BBN Systems and Technologies
A Division of Bolt Beranek and Newman Inc.

BBN Report No. 7191
Contract No. N00014-85-C-0016

RESEARCH AND DEVELOPMENT IN NATURAL LANGUAGE UNDERSTANDING AS PART OF THE STRATEGIC COMPUTING PROGRAM

FINAL REPORT

Volume 1: Overview of Technical Results

Damaris Ayuso, Madeleine Bates, Robert Bobrow, Marie Meteer, Lnace Ramshaw, Varda Shaked, Ralph Weischedel

Prepared by:

BBN Systems and Technologies
10 Moulton Street
Cambridge, MA 02138

Prepared to:

Defense Advanced Research Projects Agency (DARPA)
1400 Wilson Blvd.
Arlington, VA 22209

ARPA Order No. 5257

Contract No. N00014-85-C-0016

Scientific Officer:
Dr. Allan R. Meyrowitz

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 7191	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION BBN Systems and Technologies	6b. OFFICE SYMBOL (If applicable)
6c. ADDRESS (City, State, and ZIP Code) 10 Moulton Street Cambridge, MA 02138	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research 7b. ADDRESS (City, State, and ZIP Code) Department of the Navy Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency	8b. OFFICE SYMBOL (If applicable) DARPA
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22209	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-85-C-0016 10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Research and Development in Natural Language Understanding as Part of the Strategic Computing Program - Final Report, Vol. 1: Overview of Technical Results

12. PERSONAL AUTHOR(S) Damaris Ayuso, Madeleine Bates, Robert Bobrow, Marie Meteer, Lance Ramshaw, Varda Shaked, Ralph Weischedel

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Report	FROM 12/84 TO 9/89	1989, October revised 1990, October	85

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Natural Language processing, knowledge acquisition, discourse ill-formed input, discourse reference

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This is volume one of a three volume final report. This volume, Volume 1, provides a technical overview of the effort. Chapter 1 is the Executive Summary. Chapter 2 overviews the effort as a whole, reporting on 1) Our Integration of software and results from other Strategic Computing contractors, 2) Our participation in Fleet Command Center Battle Management Program, 3) Distribution of the resulting software to other sites, and efforts, and 4) Contributions of the work to the state of the art.

The problem of acquiring linguistic knowledge is the chief obstacle to widespread use of natural language technology. Chapters 3 and 6 report results of five to tenfold increase in our productivity in moving the natural language shells to new application domains. The ability of natural language systems to cooperatively handle novel, errorful, or incomplete forms is also critical; chapters 5 and 7 report new techniques to intelligently and gracefully respond to such forms. Chapter 4 reports on an implementation of a discourse module for understanding definite reference.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-85-C-0016. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U S. Government.

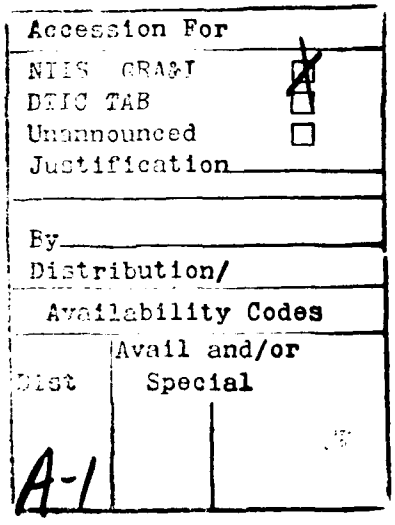

Table of Contents

[17]We would like to thank Lance Ramshaw for his invaluable help in understanding the inner workings of RUS and suggestions for where it could be augmented for our purposes, and Dawn MacLaughlin for her implementation of Parrot, the initial version of our paraphraser. We would also like to thank Ralph Weischedel, Damaris Ayuso, and David McDonald for their helpful comments on drafts of this paper and Lyn Bates for early inspirations.

List of Figures

List of Tables

1. Executive Summary

Duration: December 21, 1984 - September 30, 1989

Brief Summary of Objectives: There are three objectives of the contract:

- to perform research and development in parallel parsing, semantic representation, ill-formed input, discourse, and linguistic knowledge acquisition,
- to integrate software components from BBN and elsewhere to produce Janus, DARPA's New Generation Natural Language Interface, and
- to demonstrate state-of-the-art natural language technology in DARPA applications.

The following software has been distributed: the Janus natural language system; IRACQ, a knowledge acquisition system; system components and knowledge bases of Janus; the KL-TWO knowledge representation and inference system; various Janus components transferred to DARPA's Spoken Language Systems Project at BBN.

Summary of accomplishments:

1. The first 20 months of the effort were devoted to technology transfer of IRUS, the understanding component of Janus to the Government. To this end, BBN delivered IRUS, its knowledge acquisition tools (IRACQ and KREME), and knowledge bases for lexical semantics, lexical syntax, a domain model, and transformation rules to data base structure, to Texas Instruments for integration in DARPA's Fleet Command Center Battle Management Program (FCCBMP). Working with the Naval Ocean Systems Center, BBN officially demonstrated IRUS-86 in summer 1986 as part of the FCCBMP to representatives of CINCPACFLT, DARPA, SPAWAR, and NOSC. As of August, 1986, all of the components were transferred to NOSC, and BBN began focus on the component research goals and on the system integration goals of Janus.

2. In conjunction with the USC/Information Sciences Institute, USC/ISI's Penman language generation component was integrated with IRUS, to provide paraphrases and answers in English. This was demonstrated in May, 1987.

3. In the second version of Janus, the Penman generation component was replaced by the Mumble-86 grammar for generation from the University of Massachusetts and BBN's Spokesman text planner. This was delivered to Lockheed for integration with DARPA's AirLand Battle Management Program in late 1987. We estimate that this second version of Janus had greater functionality (e.g., not just paraphrase and answer generation, but also multi-paragraph, multi-page output generation), and that the generator was several times faster than the first version.

4. Our component research made several direct contributions of technology to BBN's Spoken Language Systems effort, including the initial grammar (a unification-based grammar), the semantic representation language, a parallel parsing algorithm, and components for mapping from the semantic representation (an intensional logic) to code for one or more application systems. In addition, we are in process of publishing our research results on dealing with errorful, novel, or unclear language; clarification dialogue; centering algorithms for reference resolution; knowledge acquisition; and a hybrid semantic representation based on intensional logic and a terminological knowledge representation.

5. Our design for seamless interfaces has been adopted in the human-machine interface for DARPA's CASES expert system.

6. Software resulting from our research and development effort has now been distributed (for R&D purposes) to the FCCBMP at the Pacific Fleet Command Center, NOSC, the University of Pennsylvania, the University of Massachusetts, Harvard University, USC/ISI, Texas Instruments, Lockheed Austin Division, RADC, and NSA.

2. Summary of Effort

2.1 Purpose

There are three objectives of the contract:

- to perform research and development in parallel parsing, semantic representation, ill-formed input, discourse, and linguistic knowledge acquisition.
- to integrate software components from BBN and elsewhere
- to produce Janus, DARPA's New Generation Natural Language Interface, and to demonstrate state-of-the-art natural language technology in DARPA applications.

Our activities as *integrator* for DARPA natural language interface work included coordinating and integrating into unified applications the work of a number of different contractors. This integration work is discussed in detail in Section 2.2.

Our natural language software was integrated with the ALBM and FCCBMP applications. We have also supported application development by building specific software to fill gaps in the integration effort. This is discussed further in Section 2.3.

We have also supported the DARPA community by distributing software to enable others' research to proceed more rapidly and cost-effectively. We discuss software dissemination for use at other sites in Section 2.4.

Finally, we conducted research on carefully focused topics in natural language. We summarize our major research achievements in parallel algorithms, robustness, the development of seamless interfaces, and integrated generation and understanding in Section 2.5.

2.2 Software Integration

BBN's responsibilities as prime contractor for DARPA's New Generation NL System are depicted in the DARPA/ISTO slide shown in Figure 2. Our task has been to integrate work from three sites having DARPA contracts: USC/Information Sciences Institute (ISI), the University of Massachusetts (UMass), and the University of Pennsylvania (UPenn). From ISI, we integrated the Penman text-generation system to BBN's IRUS-II system to create the Janus system. The ability of this system to paraphrase understanding of an input and provide English responses to queries was demonstrated at the May 1987 DARPA workshop held in Philadelphia, PA. The integrated system shared syntactic information in the lexicon and a NIKL domain model.

We integrated the DARPA supported Mumble generation grammar [34] from the University of Massachusetts with both the Spokesman generation system of DARPA's AirLand Battle Management Program (ALBM) and with BBN's IRUS-II system to form the second version of Janus. The ALBM version of Janus can generate paragraphs of the OPORD (operations order) produced by military staff using the ALBM workstation to generate battle plans. The Spokesman generation system shares the syntactic information, the domain model, and the intensional logic of the Janus understanding system.

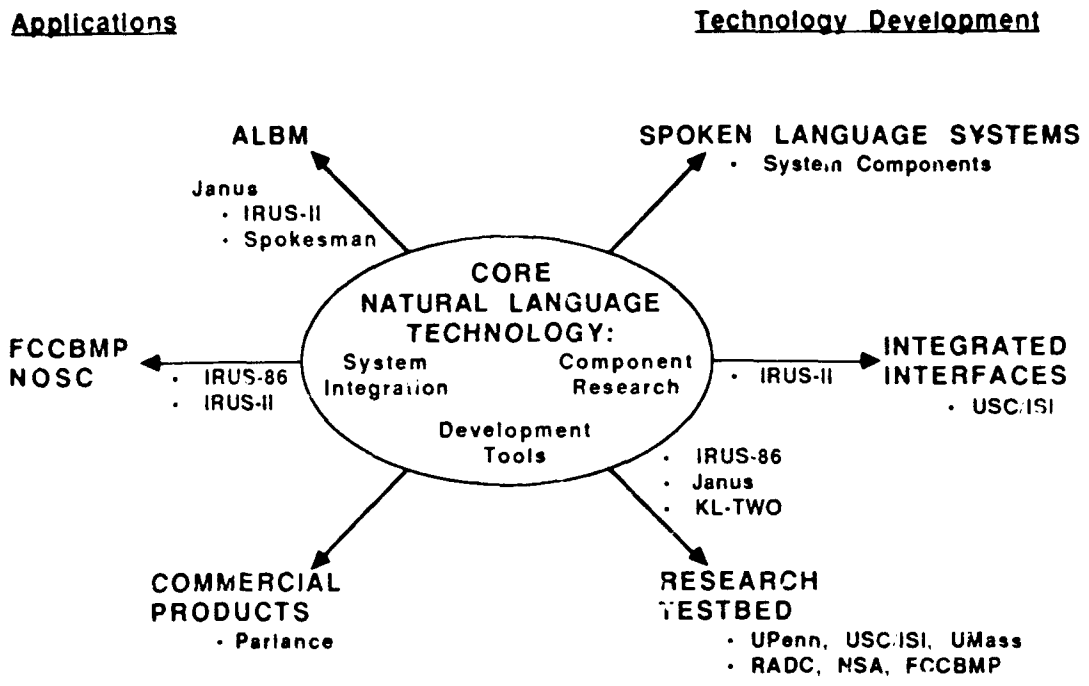

Figure 1: Transfer of BBN Core Technology for Natural Language R&D

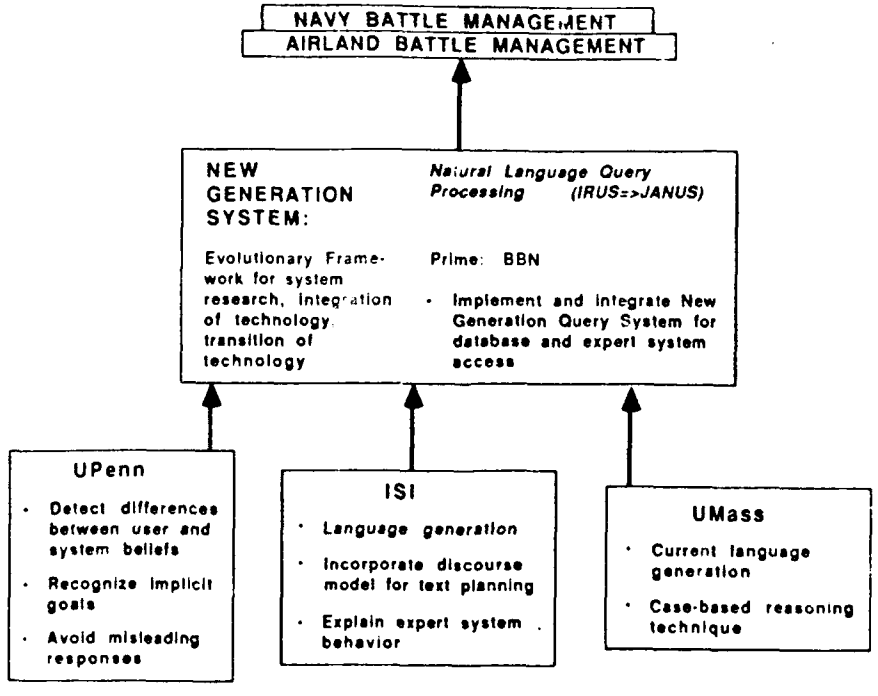

Figure 2: BBN's Role as Integration Contractor

From the DARPA work at the University of Pennsylvania we have integrated algorithms developed by Webber [50] for determining what entities can be referred to by subsequent references. We have generalized Webber's algorithm to allow the designation of referents, both by pointing (with a mouse) and natural language (with words), and to use intensional logic. We have also implemented several classes of cooperative responses [32] in the sense developed by UPenn researchers.

2.3 DARPA Application Support

Our work has supported two DARPA Strategic Computing applications: the Fleet Command Center Battle Management Program (FCCBMP) and the AirLand Battle Management Program (ALBM).

BBN's original Strategic Computing proposal promised only a short technology demonstration and evaluation of natural language as part of the FCCBMP, but a much more ambitious evaluation proved possible. BBN delivered a version of IRUS to the Naval Ocean Systems Center (NOSC) in the summer of 1985, and with help from BBN, NOSC specified information necessary to expand the domain-dependent knowledge bases of IRUS. DARPA directed BBN to curtail its contribution to FCCBMP after an August 1986 demonstration of the skeletal command and control interface developed for FCCBMP.

In late 1988, we were invited to demonstrate Janus at the FCCBMP in the Fleet Command Center, Pearl Harbor, Hawaii. Janus was demonstrated in the context of the CASES system (Capabilities Assessment Expert System), one of DARPA's FCCBMP programs. At that time, Janus demonstrated multi-modal i/o, including natural language, tables, maps, and graphics. The primary service being provided was to simplify the selection of assets to be employed to solve a CASES simulation problem. One of the achievements in that demonstration was the ability to create 90% of the 2000 domain-specific vocabulary items with only two person-days of effort. We estimate that the same capabilities would have taken at least 20 person-days with conventional tools for constructing knowledge bases.

The experience gained in transitioning the technology to NOSC and installing the system at the FCCBMP has increased BBN's understanding of priorities for NL research. It has become clear that the largest payoffs in end-user ability would result from progress in simultaneous access to several underlying systems utilities, increased robustness, faster, cheaper knowledge acquisition, the exploitation of discourse phenomena, and speech input.

In ALBM, the understanding components of Janus were demonstrated in April, 1988 as an interface to an expert system for planning unit movement, to a wargamer, to graphics displays, to knowledge bases, and to a menu hierarchy. The Janus generation component Spokesman was delivered and used to generate text for the operations order which describes the plan that the staff develops using ALBM. Spokesman was delivered as part of ALBM in 1989. Its text output supplements the map as the description of the corps-level-plan.

2.4 Software Distribution

Our software supported research and development efforts at many sites. The sites and dates on which BBN software was delivered are summarized in Table2-1.

2.4.1 Other DARPA Contractors

	IRUS	KL-TWO	Janus
FCCBMP			
NOSC	1985		1988
SAIC	1985		
TI	1985		
FCCBMP Testbed	1986		1988
ALBM			
Lockheed			1987
R&D Centers			
USC/ISI	1985	1987	1987
UPenn	1986	1987	1990
UMass	1986		
Harvard		1988	
NSA	1986	1986	
RADC	1987		
IIT			1990

Table 2-1: Software Delivered by BBN to Other Organizations

Our past collaboration with USC/ISI has been very extensive. The integration of the Penman text generation system into Janus resulted in a common lexicon supporting both generation and understanding [16, 27] and a common domain model. An interesting side result was the generality forced on both Penman and IRUS-II (BBN's understanding component in Janus). As a result of our collaboration, both Janus and Penman proved general enough to support either an intensional logic or a first-order logic as the representation of the semantics of natural language. The text planner of Penman employs KL-TWO. The inference algorithms of KL-TWO are used to reduce the logic formula into a simplified form, to determine how to break up a logic formula into subformulas that are expressible via words in the dictionary, and to find words to express each subformula.

At the University of Pennsylvania, at least six students have benefited from the availability of our systems in their graduate work [29]. In one student's work, IRUS, TEXT [33], and Mumble [34] were coordinated to illustrate UPenn's research. The grammar, parser, lexicon, semantic interpreter, and discourse components of IRUS were used to process questions to describe, compare, and contrast concepts; the output was provided to TEXT, which planned how to answer the question. The text plan was then passed to Mumble, which produced paragraph-length explanations. A second student, Brant Cheikes, is currently working on a dissertation which makes use of KL-TWO and ideas from IRUS. Cheikes is developing a unified approach to cooperative response and is using KL-TWO for knowledge representation, as well as MRL, the semantic representation language of BBN's IRUS.

2.4.2 DARPA Spoken Language Systems Research

We have supported the development of the following initial components and formalisms for DARPA's Spoken Language Systems effort at BBN:

- The unification parser and grammar. The unification parser and grammar have become the first natural language components fully integrated with BBN's speech recognition system.

- The semantic representation language for representing the meaning of a speaker's utterance. The semantic representation language, an intensional logic, is the foundation of all semantics in BBN's Spoken Language System, HARC.
- The "backend" components for translating the meaning of the speaker's utterance into code to be executed to fulfill the user's need. The initial backend came directly from the Janus demonstration of May 1987. It translates formulas in the intensional logic into database queries.
- The discourse component, which identifies the meaning of a referring expression. Pronouns (e.g., *it*, *they*, *this*, etc.), definite noun phrases (e.g., *those units*, *the three fastest subs*, etc.), and pointing are all understood using the same algorithm.

2.5 State of the Art in 1984 and How This Project Advanced It

Our effort began at the end of 1984. At that time, there was only one NLP product, INTELLECT [23, 24], which provided English access to relational data bases on mainframes. To our knowledge, there had been no installations of NLP interfaces in an operational setting, though the Naval Personnel Research and Development Center had performed tests using an NL interface to the Navy Blue file in 1978 [37]. Most research in the U.S. was focussed on interfaces to allow interactive query of relational data bases. Regarding the state of the art of the technology itself, there were several areas requiring advances if natural language interfaces were to be applied broadly; these are covered in the following sections.

2.5.1 Unknown Words, Novel Forms, and Ill-Formed Input

An interface should be forgiving of a user's deviations from its expectations, be they misspellings, typographical errors, unknown words, poor syntax, incorrect presuppositions, fragmentary forms or violated selection restrictions. Empirical studies show that as much as 25% of the input to data base query systems is ill-formed.

These capabilities have major implications for the control of the understanding process, since considering such possibilities can exponentially expand the search space. Maintaining control will require care in integrating the ill-formedness capability into the rest of the system, and also making maximal use of the guidance that can be derived from a model of the discourse and user's goals to constrain the search.

Though several general strategies for dealing with input containing unknown words, novel constructions, or errors had been proposed [11, 19, 52], the resulting search space would have been prohibitive unless the domain naturally restricted the grammar and semantics substantially. We sought ways to make dealing with ill-formed input more generally tractable.

As an example of novelty, suppose that the word *preparedness* is known to the interface but that its use as an attribute of a ship in a sentence such as *What is the preparedness of Assurance?* is novel. Our work demonstrates the potential of inferring such new meanings from context. If the sentence in question occurred in the context of assigning assets to locate a possible enemy submarine, *preparedness* would most likely mean the ASW (anti-submarine warfare) readiness rating, and not any of dozens of other properties (including other types of readiness) of a surface vessel such

as Assurance. We have demonstrated the ability to infor a meaning or present a ranked list of options to the user, rather than be blocked by such everyday novelty.

Not even humans understand every utterance. When an input cannot be understood, asking a clarifying question about what is not understood or explaining why the system does not understand is essential to continuing communication. BBN has begun research [35] to permit the system to paraphrase its understanding of the user's meaning or explain what it did not understand. The system does not have a model of metonymy adequate to understand the figure of speech used in the utterance *Which commanders are assigned to problem2?*. The system's model of the domain may allow for assignment of vessels, but not people, to a problem. Janus diagnoses the probable cause of its misunderstanding, plans how to respond, and generates the explanation *I don't understand how commanders can be assigned*. BBN has developed a model for understanding novel language, performance errors, vagueness, and ambiguity [52, 53, 59].

Full solutions to many classes of these phenomena remain to be worked out in detail, but our approaches to dealing with novel/ambiguous language and to responding appropriately to input that cannot be understood are quite promising. While we have demonstrated the ability of such approaches using Janus, effort will be required to prepare these approaches for transfer to applications.

In summary, we have developed two new techniques:

- Applying a model of user goals and problem-solving behavior to predict what an unknown word means in context.
- Offering an explanation why an input could not be understood.

For greater detail, see [35, 41, 42, 54] or chapter five and seven of this volume.

2.5.2 Parallel Processing

While sequential algorithms could process well-formed typed text rapidly enough in 1984, to deal with ill-formed input or to process speech where many possible sequences of words would need to be considered, parallel processing seemed highly desirable. Furthermore, parallel hardware from various vendors was clearly visible on the horizon. Little work in parallel NLP had been attempted.

The challenge in this is that optimization strategies in sequential algorithms has tended to introduce side-effects. For instance, without doubt, the augmented transition network was most frequently used to build NLP systems [40] because it permitted the introduction of side-effects and optimization. The RUS grammar, our large coverage grammar of English, crucially uses both for optimization and simplification of the grammar.

A core problem in natural language processing is *parsing*, the identification of which words form elemental phrases and the recognition of how those phrases combine to form larger phrases based on a grammar. Parsing is one of the best-understood aspects of natural language processing. Much of current research in grammar is based on a cluster of grammar formalisms called *unification grammars*; a *unification parser* parses sentences using such a grammar [45]. A parallel unification parser can have broad applicability to the next generation of natural language processors because it allows the latest research developments to be implemented on parallel machines.

Our research results on parallel parsers [21] are summarized below:

1. We defined a subclass of unification grammars for which the parser is guaranteed to halt[1] and which seem particularly well-suited to natural language constraints.

2. We generalized and parallelized the Cocke-Kasami-Younger algorithm for context free grammars to produce a parallel parser for this subclass of unification grammars.

3. We implemented a preliminary, experimental version of the algorithm in a parallel version of Lisp, initially obtaining a five-fold speedup on a 16 node parallel processor. (Considerable further speedup will be possible.)

4. We produced a grammar for a substantial subset of English in the formalism. This grammar is now at the heart of BBN's DARPA-funded Spoken Language System.

For technical detail, see [21, 22, 55].

2.5.3 Discourse Processing

The meaning of a sentence depends in many ways on the context which has been set up by the preceding discourse. In 1984, NLP systems employed a rather shallow model of discourse structure. The most obvious impact of context is in the interpretation of referring expressions, i.e., pronouns definite and pointing.

Our work in discourse focussed on interpreting, referring expression correctly. Discourse entities (DEs) are descriptions of objects, groups of objects, events, etc. from the real world or from hypothesized or possible worlds that are evoked in a discourse; any of these may be referred to in later discourse. Any communicative act, be it spoken, written, gestured, or system-initiated, can give rise to DEs. As a discourse progresses, an adequate discourse model must represent the relevant entities, and the relationships between them [20]. A speaker may then felicitously refer anaphorically to an object if there is an existing DE representing it, or if a corresponding DE may be directly inferred from an existing DE. For example, the utterance *Every senior in Milford High School has a car* gives rise to at least 3 entities, describable in English as *the seniors in Milford High School*, *Milford High School*, and *the set of cars each of which is owned by some senior in Milford High School*. These entities may then be accessed by the following utterances, respectively:

They graduate in June.
It's a good school.
They completely fill the parking lot.

We integrated into Janus a strategy developed by Webber of the University of Pennsylvania [49, 51]. This is, to our knowledge, the first implementation of Webber's DE generation ideas. We designed the algorithms and structures necessary to generate discourse entities from our logical representation of the meaning of utterances, and from pointing gestures, and currently use them in Janus's [58, 6] pronoun resolution component, which applies syntactic and semantic constraints to track and interpret references.

Webber's general approach to discourse entity generation from a logical representation proved very useful in our efforts. We were able to recast her basic ideas in our logical framework, and currently use the generated DEs extensively.

A significant extension to her work is our technique for handling pointing by generating discourse entities which are then used in the pronoun resolution component

[1] An unrestricted unification formalism is equivalent to a Turing machine, and, therefore, a parsing algorithm for the full class of unification grammars is not guaranteed to halt on all input.

uniformly with the others. For example, after the request *Show me the C1 carriers in the Indian Ocean* the system will display icons on the color monitor representing the carriers. The user can then say *Which of them are within 200 miles of it? <point with mouse to Kennedy>.*

The fact that the generation of DEs is done via structural rules operating on a semantic representation provided a degree of modularity that allowed our pronoun resolution component to work *automatically* when we combined a new syntactic component with our semantic and discourse component (replacing an ATN by a unification grammar. The discourse component has been ported to the BBN Spoken Language System [8]. The fact that entity representations are mostly semantic in nature, not syntactic, also facilitated the addition and use of non-linguistic entities in a *uniform way*.

Our paraphrasing component [35] already uses the discourse entities in generating English answers and English explanations. One area of future work is to have the language generator make more extensive use of them, so it can smoothly refer to focused objects.

In summary, we have integrated ideas on discourse processing from the University of Pennsylvania, implemented then, extended them to handle references made by pointing as well as verbal references, and employed them not only to interpret referring expressions but also to generate referring expressions. For technical detail, see chapter four of this volume or [4].

2.5.4 Semantic Interpretation and Knowledge Representation

The two most-often used semantic representation languages in NLP as of 1984 were first-order logic and frame languages which are loosely equivalent to a prepositional logic. Though we had found first-order representations adequate (and desirable) for NL interfaces to relational data bases, we felt a richer semantic representation was important for future applications. The following classes of representation challenges motivated our choice.

- Explicit representations of time and world. Object-oriented simulation systems were an application that involved these, as were expert systems supporting hypothetical worlds. The underlying application systems involved a tree of possible worlds. Typical questions about these included *What if the stop time were 20 hours?* to set up a possible world and run a simulation, and *In which situations is blue attrition greater than 50%?* where the whole tree of worlds is to be examined. The potential of time-varying entities existed in some of the applications as well, whether attribute values (as in *How often has USS Enterprise been C3?*) or entities (*When was CV22 decommissioned?*) The time and world indices of our intensional logic WML provided the opportunity to address such semantic phenomena (though other logics might serve this purpose).

- Distributive/collective quantification. collective readings could arise, though they appear rare, e.g., *Do USS Frederick's capabilities include anti-submarine warfare* or *When did the ships collide?* See[25] for a computational treatment of distributive/collective readings in WML.

- Generics and Mass Terms. Mass terms and generally true statements arise in these applications, such as in *Do nuclear carriers carry JP5?*, where JP5 is a kind of jet fuel. Term-forming operators and operators on predicates are one approach and can be accommodated in intensional logics.

- Propositional Attitudes. Statements of user preference, e.g., *I want to leave in the afternoon*, should be accommodated in interfaces to expert systems, as should statements of belief, *I believe I must fly with a U.S. carrier*. Since intensional logics allow operators on predicates and on propositions, such statements may be conveniently represented.

Our second motivation for choosing intensional logic was our desire to capitalize on other advantages we perceived for applying it to natural language processing (NLP), such as the potential simplicity and compositionality of mapping from syntactic form to semantic representation and the many studies in linguistic semantics that assume some form of intensional logic.

However, the disadvantages of intensional logic for NLP include:

- The complexity of logical expressions is great even for relatively straightforward utterances using Montague grammar [21].

- Real-time inference strategies are a challenge for so rich a logic.

In order to gain the increased expressive power of intensional logic (with its simplicity and compositionality of mapping from syntactic form to semantic representation) while overcoming the computational drawbacks of reasoning algorithms over an intensional logic, Janus uses a hybrid approach to representation. The meaning of an utterance is represented as an expression in WML (World Model Language) [25], which is an intensional logic. However, a logic merely prescribes the framework of semantics and of ontology. The *descriptive constants*, that is the individual constants (functions with no arguments), the other function symbols, and the predicate symbols, are abstractions without any detailed commitment to ontology. An intensional logic is the representation language for the semantics of utterances; a frame-based language specifying of descriptive constants and axioms relating them in a given application domain is employed for all reasoning.

Axioms stating the relationships between the constants are defined in NIKL [9, 38]. We wished to explore whether a language with limited expressive power but fast reasoning procedures is adequate for core problems in natural language processing. The NIKL axioms constrain the set of possible models for the logic in a given domain.

Though we have found clear examples that argue for more expressive power than NIKL provides, 99.9% of the examples in our expert system and data base applications have fit well within the constraints of NIKL. Based on our experience and that of others, the axioms and limited inference algorithms can be used for classes of anaphora resolution, interpretation of highly polysemous or vague words such as have and with, finding omitted relations in novel nominal compounds, and selecting modifier attachment based on selection restrictions.>

Hybrid representation systems have been explored before [10, 44, 48], but only the experience developed in Janus is based on an extensive natural language processing system.

For technical detail, see [25, 56].

2.5.5 Seamless Interfaces

As the complexity and number of application systems available to a user in a given computing environment increases, understanding the idiosyncrasies of each system becomes a significant burden to users. The imposition of a common appearance and functionality begins to address this problem. However, the consistency imposed can become a frustrating rigidity for experienced users. A major research effort at BBN since 1986 has been to address the problem of providing the user with unified access to all available resources by allowing flexible use of natural language, menus, graphics, and pointing. The goal of this work is to achieve seamlessness along the following dimensions:

- **Seamlessness between modalities**: The user should be able to mix input modalities freely and switch between modalities in a natural way during a session.
- **Seamlessness between underlying systems**: The user should not have to be consciously aware of distinctions between underlying systems; a query should be routed appropriately (and decomposed if necessary).
- **Seamlessness between experience levels**: Users with different purposes or experience levels should be able to access the functionality appropriate to their point of view and experience.

BBN has been able to demonstrate great advances in the area of seamless interfaces. Janus integrates natural language understanding with a sophisticated graphics interface and with an NL generator. This seamless interface was demonstrated in the domain of prosecuting SPAs, in December, 1988 at the FCCBMP testbed in Hawaii. The SPA demonstration allows the combining of natural language with pointing to graph, map, and table entities to yield a very natural interface. Pointing at a screen icon introduces a discourse entity [50], and references to it are resolved uniformly with other forms of reference. The same request can be made via menus, pointing, or natural language.[2]

Our approach has since been generalized to use declarative descriptions of each system's functionality. From the meaning representation of a query and the system descriptions, an execution plan is derived which accesses the appropriate systems and composes a response. This approach has been successfully demonstrated in the AirLand Battle Management Program (ALBM), in April, 1988 and in the Fleet Command Center Battle Managment Program in December, 1988.

In the demonstration for ALBM, natural language access was provided to data in Intellicorp's Kee[3] system, to objects representing hypothetical worlds in an object-oriented simulation system, and to Lisp functions capable of manipulating this data. In the FCCBMP, a natural language was provided to access as CASES subset of the Integrated Data Base and a set of CASES functions modelling Navy problems; this demonstration provided access and control to more than 800 functions related to Navy problem-solving.

In sum, our work in seamless interfaces provides highly desirable utility along the following two dimensions:

- It frees the user from having to identify for each term (word) pieces of program that would carry out their meaning.
- It improves the modularity of the interface, insulating the presentation of

[2]A videotape demonstrating this system and its use is available.

[3]KEE is a trademark of Intellicorp Inc.

information, such as table i/o, form details fo the underlying application(s).

For additional technical detail, see [4, 7, 43].

2.5.6 Integrating Generation and Understanding

One of the most dramatic results already demonstrated in Janus is the combining of general-purpose, linguistically sound, extensible understanding and generation in a single system. The technology for generating natural language expressions, e.g., as a response to a question, in 1984 was so distinct from the technology for understanding natural language, that only one system [26] had attempted to combine both technologies in a single system. That effort was for processing German; only the most rudimentary components were shared between the generation subsystem and the understanding subsystem.

One novel aspect of our work has been the emphasis on driving both the understanding and generation from the same knowledge bases. In integrating USC/ISI's Penman text generation system into the first version of Janus, we learned how to share the lexicon and domain model, two of the four knowledge bases. In integrating Mumble-86 from the University of Massachusetts into the second version of Janus, we have now modified the third of these knowledge bases, which will allow the same knowledge of the semantics of words and phrases to drive both understanding and generation.

For technical detail, see [16, 31, 35] and Parts II and III of this volume.

2.5.7 Knowledge Acquisition

Although natural language systems have achieved a high degree of domain independence through separating domain-independent modules from domain-dependent knowledge bases, portability, as measured by effort to move from one application domain to another, has been a problem. The cost in person effort to achieve some pre-specified degree of coverage in a new application domain is high for all natural systems. To port to a new application domain, say a data base of readiness, positional, and technical data regarding Navy units rather than a data base on logistics, requires acquisition of basic facts about a domain. Examples include the following:

- Basic facts, for instance, all vessels are units and platforms and all vessels have an overall combat readiness rating. Such facts state the basic classes of entities in the domain, defined subclasses of entities, and relations among the defined classes.
- Syntactic and semantic facts about the domain vocabulary. For instance, *CROVL* is a noun, has no plural form, and refers to the overall combat readiness rating of a vessel.
- Facts about the underlying system. In order to be able to answer a question about whether a vessel is equipped with harpoon missiles, the system must know that the information is in the HARP.CAPABILITY field of the UCHAR table, and that that field holds "Y" if the vessel is so fitted and "N" if not.

In 1984, the only way to provide such facts was by painstaking handcrafting of such knowledge. Our experience in installing our natural language interface as part of DARPA's Fleet Command Center

Battle Management Program (FCCBMP) illustrates the kind of portability needed if NL applications (or products) are to become widespread. We demonstrated broad linguistic coverage across 35 fields of a large Oracle database, the Integrated Data Base (IDB), in August 1986.

We had developed a suite of tools to greatly increase our productivity in porting IRUS, the predecessor to the Janus NL understanding and generation system to new domains. KREME [1] enables creating, browsing, and maintaining of taxonomic knowledge bases. IRACQ [3] supports learning lexical semantics from examples with only one unknown word. Both of those tools were used in preparing the FCCBMP demonstration in 1986. However, these do not sufficiently reduce the human effort involved in building the knowledge bases.

As an illustration of this, consider work at the Naval Ocean Systems Center with the IRUS system. Figure 3 summarizes the measurement of their performance using BBN's state-of-the-art tools. The graphs show slow but steady progress in defining four knowledge bases:

- Lexical syntactic and morphological information
- Application mapping rules which tell how to map a symbol in the logical representation of the user's input into the structures of the underlying system
- (Lexical) semantic interpretation rules for semantic classes of words, idioms and phrases
- The symbols (logical constants) corresponding to concepts and relations of the user's view of the domain and application.

It is estimated that the utility, as measured by number of fields accessible in the Integrated Data Base (IDB), grew 35 to 57 from August 1985 to December 1987. That represents 17% of a grand total of approximately 330 total fields at the time.

While clear progress was present, far more rapid progress is not just desirable but necessary, and would substantially reduce the cost of making natural language technology available in applications. *What was missing was a way to rapidly infer the knowledge bases for the overwhelming majority of words used in accessing fields.* If such a tool were available, then one could further bootstrap using IRACQ.

As a result of this we started two efforts to address the need for more productive knowledge acquisition. The following two subsections describe that work.

2.5.7.1 Increasing Productivity by More Powerful Tools

KNACQ [57] serves this purpose. A frame-based domain model is used to organize, guide, and assist in acquiring the syntax and semantics of domain-specific vocabulary. Using the browsing facilities, graphical views, and consistency checker of KREME [1] on NIKL taxonomies, one may select any concept or role for knowledge acquisition. KNACQ presents the user with a few questions and menus to elicit the English expressions used to refer to that concept or role.

The information acquired through KNACQ is used not only by the understanding components and but also by BBN's Spokesman generation components for paraphrasing, for providing clarification responses, and for answers in English.

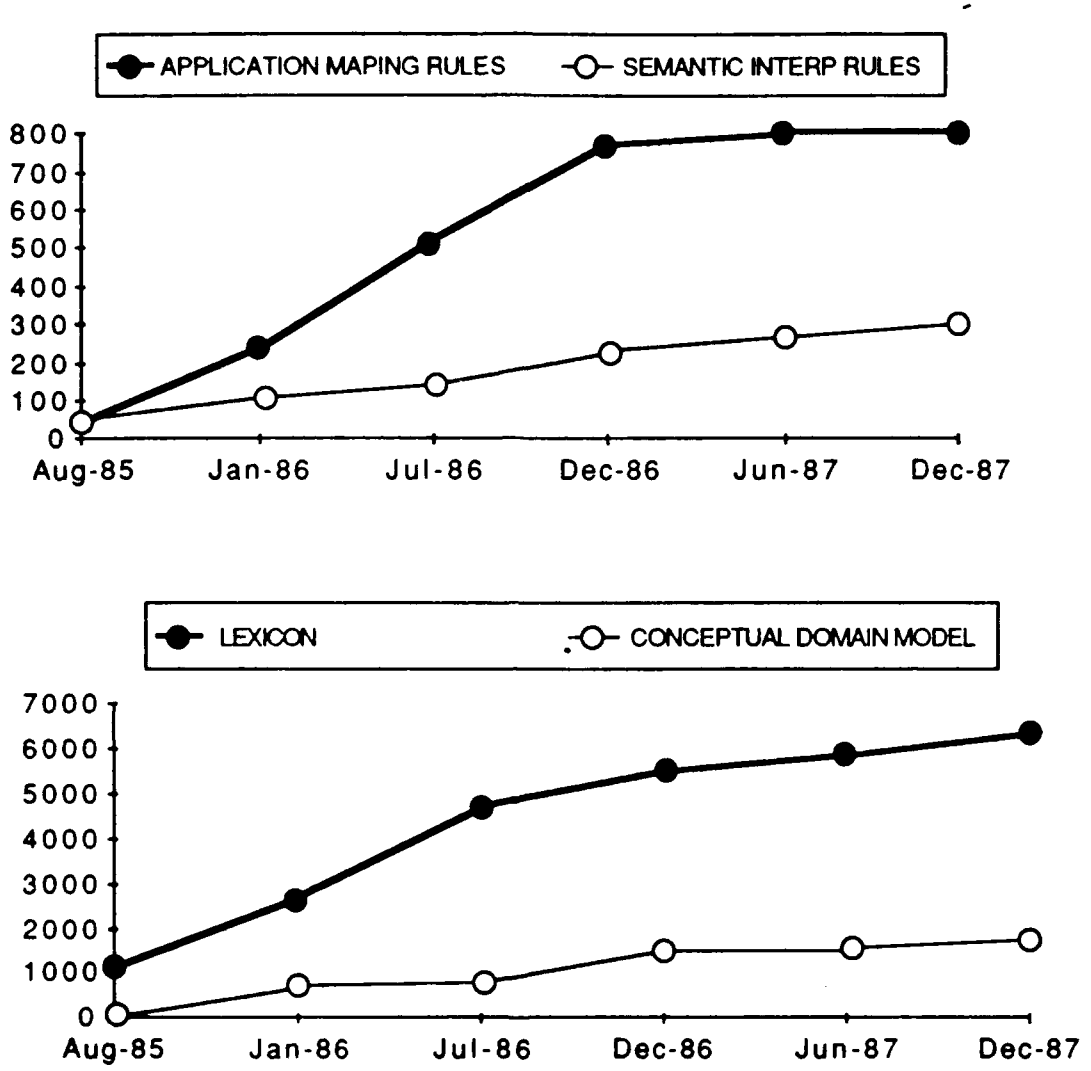

Figure 3: Growth in Knowledge Bases

There are several things we have learned even in the early stages of KNACQ's development based on porting Janus to CASES, an expert system in DARPA's Fleet Command Center Battle Management Program (FCCBMP). In this use of KNACQ, the original domain model pertinent to the portion requiring a natural language interface consisted of 189 concepts and 398 roles.

First, no restructuring of that domain model was necessary, nor was any deletion required. Second, we found it useful to define some additional concepts and roles. Certain subclasses and attributes not critical to the expert system were nevertheless lexically significant. Third, 1093 proper nouns (e.g., ship and port names) were inferred automatically from the instances in the expert system taxonomy.

As a result, the time required to supply lexical syntax and semantics was much less than we had experienced before developing KNACQ. In two days we were able to provide 563 lexical entries (root forms not counting morphological variants) for 103 concepts and 353 roles. Together with the automatically inferred proper nouns, this was approximately 91% of the domain-dependent vocabulary used for a demonstration given in the Fleet Command Center, Pearl Harbor, Hawaii in December, 1988, and interfacing to the CASES expert system. (Approximately 500 root words in addition are domain independent.) *That is about 5-10 times more productivity than we had experienced before with manual means.*

Since we had identified knowledge acquisition of domain-specific vocabulary as the critical bottleneck, a parallel effort (not under Government contract) to add a knowledge acquisition component (the Learner[TM] system[4]) was initiated for BBN's natural language interface product (Parlance[TM]).

During 1988, BBN used its Learner tool to configure the Parlance database interface to two different versions of a large Navy database. The configuration process was performed primarily with development versions of the Learner, which is a software tool for creating the knowledge bases, vocabulary, and mappings to the database that enable the Parlance interface to understand questions addressed to a particular database. The Learner reduced the time required to create Parlance configurations from months to weeks, and demonstrated that the Learner works effectively on databases with many hundreds of fields.

The Integrated Data Base (IDB) is a large, evolving database developed under DARPA's Battle Management Programs and being used in the Fleet Command Center in Pearl Harbor, Hawaii [12]. It has dozens of tables and hundreds of fields containing information about hundreds of U.S. ships, planes and other units, as well as more limited data on foreign units.

We compared this by-hand configuration process with the first experience using the Learner on the IDB. The two examples used different databases, but in each case we began with a large set of sample queries in the target domain, and periodically tested the developing configuration by running those queries through the Parlance system. We measured our progress by keeping track of the number of those queries the system could understand as the configuration process went on. This comparison actually considerably understates the productivity enhancement realized with the Learner, because the personnel database used for the by-hand configuration was much smaller and less complex than the IDB.

Overall, the new knowledge acquisition tools in acquiring domain-specific

[4]Learner and Parlance are trademarks of BBN Systems and Technologies.

knowledge by at least a factor of 10. Contrast the fact that about 9 person-weeks was neeeded to cover the 660 fields of the IDB in 1988 with the tables of figure 3 where 57 data base fields were covered after a few person years of effort.

For additional technical detail, see [3, 5, 57] or chapters three and six of this volume.

2.5.7.2 Towards a Domain-Independent Dictionary

Recently a handful of efforts have focussed on deriving large knowledge bases of common facts. (The CYC project at MCC is employing 10-20 programmers to handcraft a knowledge base based on a selection of encyclopedia articles [30]. At IBM Yorktown Heights [13, 28, 39] and Bell Communications Research, proprietary efforts are underway to automatically derive synonym sets and other information from online dictionaries.)

Our effort built on software derived from an earlier IR&D effort, the Common Facts Data Base (CFDB) [15]. CFDB has been used to derive common facts from dictionaries and is experimentally being applied to other reference material. A data base of over 500,000 tuples of common facts exists already. More importantly, the software to assimilate additional facts from reference texts is available.

Here we illustrate two of the many ways such automatically derivable data bases can increase robustness compared to today's systems. A long-standing problem is the interpretation of nominal compounds, sequences of nouns such as, *carrier task force*. Heretofore one had to handcraft a definition for each example or small class of examples. Some informal definitions are provided directly in dictionaries for frequently occurring, well-known expressions, e.g., *fire engine*. In our approach, these are representable in the CFDB and therefore are automatically derivable. Others follow regular patterns [2, 18], such as part-whole relations, which require common facts, like those in CFDB, in order to be interpreted. For example, if the NLP system encounters *helicopter rotor* for the first time, it could be understood if the CFDB contains the knowledge that a rotor is part of a helicopter.

Another long-standing problem is interpreting definite references (also called discourse anaphora). The use of syntactic information to constrain and rank what an anaphoric expression can refer to is rather well understood. References involving the same terminology are also rather well understood, e.g., using *those ships* as a short form after *Which ships are nearest location A?* What illustrates non-robustness in current discourse components are those that require a "bridge" [14] between what is mentioned, e.g., *the flight deck*, and the expression that implies its existence, e.g., *the carrier Midway*. Our hypothesis is that bridges fall into one of potentially a few dozen patterns, in this case, referring to a part after mentioning the whole. The common fact that is needed is that aircraft carriers have a flight deck. Such bridges require large volumes of common, mundane facts, such as those in the CFDB.

Both nominal compounds and discourse anaphora seem to fall into a few dozen semantic patterns, each of which assumes a large set of common facts. It has been easy to implement such semantic patterns for some time; what has been lacking is a way to automatically derive the large set of common facts assumed.

For additional technical detail, see [15]

2.6 The Architecture of Janus

The design of the Janus natural language interface supports the following desiderata:

- A natural language interface (NLI) should support access to general decision support aids, e.g., expert systems, simulation systems, forecasting tools, report generation aids, graphics capabilities, application-specific calculations, etc., not just data base access. Janus has been interfaced to relational data bases, object-oriented data bases, simulation systems, and mathematical modeling application software.
- An NLI should support a user's need to access several application systems with a single request, automatically determining a decomposition of the problem into subproblems for the individual application systems. In Janus, we have demonstrated this in both DARPA's AirLand Battle Management Program and Fleet Command Center Battle Management Program.
- Details of the underlying application system(s) are modularized in Janus so that as the underlying system evolves, only a proportional portion of the NLI should require change.
- Janus is forgiving of a wide variety of input errors, including typographical errors, unknown words, omitted prepositions, subject-verb disagreement, and cryptic forms.
- Janus is portable, minimizing effort in moving the interface to a new domain, through new knowledge acquisition tools.
- Janus supports nonlinguistic features that users expect in interfaces, including pointing, table i/o, graphs, and menus.

Since many system architectures could presumably support those goals, we do not claim that the system structure could be derived uniquely from the goals. Rather, the intent is to provide the rationale behind our design.

In general, one could view an NLI monolithically as in Figure 4.

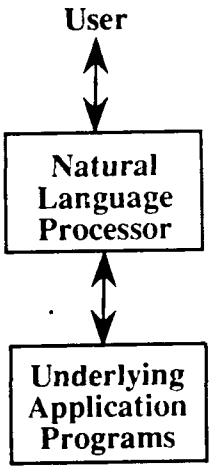

Figure 4: A Monolithic View

The first division we argue for is modularization of processing based on language properties and of properties of the underlying application system(s), as in Figure 5.

While it seems unlikely that any NLI has ever had a monolithic structure, it is not uncommon for an NLI to not modularize the underlying system structure from language processing. For instance, in TQA [17] the content of lexical entries reflected the structure and semantics of a given data base. In REL [47], the data base was assumed to be downloaded into a semantic network consistent with REL's semantic processing.

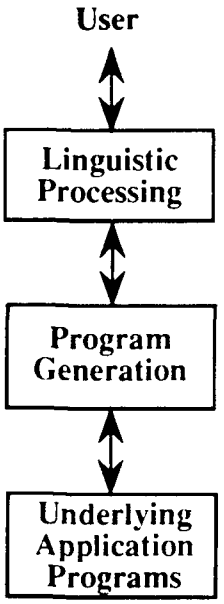

Figure 5: Separation of Language and Underlying System Structure

There are several clear advantages to modularizing the system in this way.

- As the underlying system evolves, the NLI changes are localized. For instance, consider data base management systems, the application of NLI's which has received the most attention. The whole data base could be reorganized; yet, the linguistic processing, such as syntactic, semantic, and discourse processing is not affected. Also, data base machine could replace a dbms on a conventional computer architecture; nevertheless, if the same information is in the data base, the linguistic processing is unchanged.

  In fact, we have tested this with Janus, first providing access to 45 fields of an SQL data base. When expert system capability was added to the application, the data was transformed into an object-oriented representation with Lisp routines as the primitives for access. The same functionality was achieved without a single change to the linguistic components.

- For instance, we were simultaneously involved in an expert system support for Navy planning and for Army planning. Though the implications for terms are subtly different in the two domains of resources management, they can share the same vocabulary and lexical syntax/semantics for words such as (military) unit, deploy, miles, commanding officer, hours, readiness, rating, etc.

- Furthermore, the modularization implied in Figure 5 allows one to understand and respond to user input not covered by the application system. The user may not have a precise model of the functional capabilities of the applications being accessed.

There are three other critical design decisions in Janus, in addition to the one separating linguistic processing from program generation. The most important is the choice of what to represent in data modules versus what to include in program components. Our decision was to modularize that which is application-specific as a data via encoding that information in knowledge bases. Therefore, to port Janus to a new domain, only four modules would have to change: knowledge of the syntactic properties of the words of the domain; knowledge of the semantics of the words of the domain; a domain model stating the defined classes of entities of the domain and defined relations among them; and the correspondence from classes and relations of the domain model to functions provided by the application systems. Since the domain-specific data is modularized into knowledge bases, knowledge acquisition tools were successfully designed to facilitate porting to new application domains.

Another design decision was inevitable given the state of the art in 1984. The linguistic components for understanding would be distinct from those for generation. Even now, the goal of having a single set of algorithms for both understanding and generation is a long-term research goal [46]. Consequently Janus is derived in part from earlier understanding and generation components that share the domain model, lexical knowledge, and portions of the discourse model.

The remaining fundamental design decisions relate the decomposition of the linguistic components into modules. Since there is ample evidence for distinguishing among the syntactic phenomena, the semantic phenomena, and the discourse phenomena of language, the understanding subsystem (IRUS-II) consists of intercommunicating syntactic, semantic, and discourse components. In the Spokesman generation subsystem [36], the text planning component makes decisions about what is said, producing a text plan; then the Mumble generator [34] transforms the text plan into English text.

A high-level view of the structure of Janus appears in Figure 6.

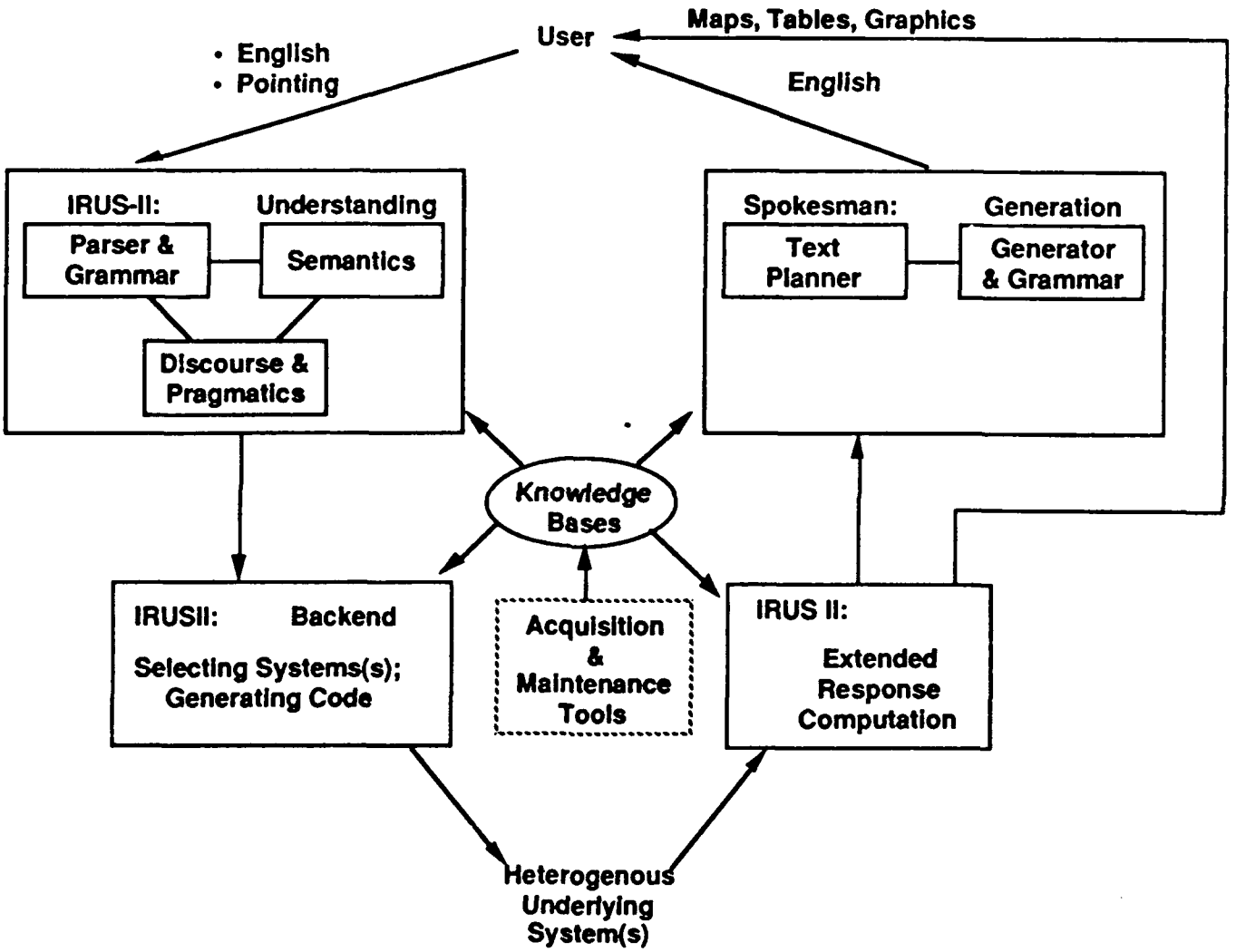

Figure 6: Structure of Janus

2.7 Lessons Learned

Two general conclusions stand out from our experience. First, the most critical to increase the utility of natural language processing is to reduce the cost it takes to port a system to a new domain. That means reducing the effort to acquire the knowledge about the semantics of the domain, about the vocabulary used in the domain, and about the relationship between the vocabulary and the underlying system(s) used in that domain. The most mature areas of natural language research, grammar formalisms and semantic formalisms, seem to already offer several adequate alternatives for NL interfaces. Therefore, far more effort should be focussed on the area of knowledge acquisition.

Second, the integration of natural language capabilities into a seamless interface would add significant additional utility. The ability to access several underlying systems to solve a problem seems critical in the next generation of command and control systems and planning systems. Such interfaces will presume state of the art capabilities in graphics, pointing, tables, and charts.

References

[1] Abrett, G. and Burstein, M. The KREME Knowledge Editing Environment. *Int. J. Man-Machine Studies* 27:103-126, 1987.

[2] Ayuso Planes, D. The Logical Interpretation of Noun Compounds. Master's thesis, Massachusetts Institute of Technology, June, 1985.

[3] Ayuso, D.M., Shaked, V., and Weischedel, R.M. An Environment for Acquiring Semantic Information. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 32-40. ACL, 1987.

[4] Ayuso, D. Discourse Entities in Janus. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 243-250. 1989.

[5] Bates, M. Rapid Porting of the Parlance[tm] Natural Language Interface. In *Speech and Natural Language*, pages 83-88. Morgan Kaufmann Publishers Inc., San Mateo, CA, 1989.

[6] BBN Systems and Technologies Corp. *A Guide to IRUS-II Application Development in the FCCBMP*. BBN Report 6859, BBN Systems and Technologies Corp., Cambridge, MA, 1988.

[7] Bobrow, R. J., Resnik, P., and Weischedel, R. M. Multiple Underlying Systems: Translating User Requests into Programs to Produce Answers. In *Proceedings of the 28th Annual Meeting of the Association for Compuational Linguistics*, pages 227-234. Association for Computational Linguistics, 1990.

[8] Boisen, S., Chow Y., Haas, A, Ingria, R., Roucos, S., Scha, R., Stallard, D., and Vilain, M. *Integration of Speech and Natural Language: Final Report*. BBN Report 6991, BBN Systems and Technologies Corp., 1989.

[9] Brachman, R.J. and Schmolze, J.G. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9(2), April, 1985.

[10] Brachman, R.J., Gilbert, V.P., and Levesque, H.J. An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of Krypton. In *Proceedings of IJCAI85*, pages 532-539. International Joint Conferences on Artificial Intelligence, Inc., Morgan Kaufmann Publishers, Inc., Los Angeles, CA, August, 1985.

[11] Carbonell, J.G. and Hayes, P.J. Recovery Strategies for Parsing Extragrammatical Language. *American Journal of Computational Linguistics* 9(3-4):123-146, 1983.

[12] Ceruti, M.G., Schill, J.P. *FCCBMP Data Dictionary, Version 3*. Technical Report, Naval Ocean Systems Center, Code 423, San Diego, CA, 1988.

[13] Chodorow, M.S., Ravin, Y., and Sachar, H.E. A Tool for Investigating the Synonymy Relation in a Sense Disambiguated Thesaurus. In *In Proceedings of the Second Conference on Applied Natural Language Processing*. 1988.

[14] Clark, H.H. Bridging. In *Theoretical Issues in Natural Language Processing*, pages 169-174. 1975.

[15] Crowther, W. A Common Facts Data Base. In *Speech and Natural Language*, pages 89-93. Morgan Kaufmann Publishers Inc., San Mateo, CA, 1989.

[16] Cumming, S. *Design of a Master Lexicon*. Technical Report ISI/RR-85-163, USC/ISI, 1986.

[17] Damerau, F.J. Operating Statistics for the Transformational Question Answering System. *American Journal of Computational Linguistics* 7(1):30-42, 1981.

[18] Finin, T.W. The Semantic Interpretation of Nominal Compounds. In *Proceedings of The First Annual National Conference on Artificial Intelligence*, pages 310-312. The American Association for Artificial Intelligence, The American Association for Artificial Intelligence, August, 1980.

[19] Granger, R.H. The NOMAD System: Expectation-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-Formed Text. *American Journal of Computational Linguistics* 9(3-4):188-198, 1983.

[20] Grosz, B., and Sidner, C. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics* 12(3):175-204, July-September, 1986.

[21] Haas, A. Parallel Parsing for Unification Grammars. In *Proceedings of the International Joint Conference on Artificial Intelligence*. AAAI, 1987.

[22] Haas, A. A Parsing Algorithm for Unification Grammar. *Computational Linguistics* 15(4), December, 1989.

[23] Harris, L.R. User Oriented Base Query with the ROBOT Natural Language Query System. *Int. Journal of Man-Machine Studies* 9:697-713, 1977.

[24] Harris, L.R. The ROBOT System: Natural Language Processing applied to Data Base Query. In *Proceeding 1978 Annual Conference*, pages 165-172. Association for Computing Machinery, Washington, D.C., December, 1978.

[25] Hinrichs, E.W., Ayuso, D.M., and Scha, R. The Syntax and Semantics of the JANUS Semantic Interpretation Language. In *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program, Annual Technical Report December 1985 - December 1986*, pages 27-31. BBN Laboratories, Report No. 6522, 1987.

[26] Hoeppner, W., Christaller, T., Marburger, H., Morik, K., Nebel, B., O'Leary, M., and Wahlster, W. Beyond Domain-Independence: Experience with the Development of a German Language Access System to Highly Diverse Background Systems. In *Proceedings of the Eighth Interantional Joint Conference on Artificial Intelligence*, pages 588-594. International Joint Conference on Artificial Intelligence, 1983.

[27] Ingria, R. Lexical Information for Parsing Systems: Points of Convergence and Divergence. *Automating the Lexicon.* , . To appear.

[28] Jensen, K. and Binot, J-L. Dictionary Text Entries as a Source of Knowledge for Syntactic and Other Disambiguations. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 152-159. 1988.

[29] Joshi, A.K. personal communication. 1988

[30] Lenat, D., Prakash, M., and Shepherd, M. CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. *AI Magazine* 6(4):65-85, 1986.

[31] MacLaughlin, D. *Parrot: The Janus Paraphraser*. BBN Report 7139, BBN Systems and Technologies Corp., Cambridge, MA, 1989.

[32] Mays, E. Failures in Natural Language Systems: Applications to Data Base Query Systems. *Proceedings of the National Conference on Artificial Intelligence* , 1980.

[33] McKeown, K.R. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, Cambridge, England, 1985.

[34] Meteer, M., McDonald, D., Anderson, S., Forster, D. Gay, L., Heuttner, A., Sibun, P. *Mumble-86: Design and Implementation*. Technical Report 87-87, University of Massachusetts, September, 1987.

[35] Meteer, M. and Shaked, V. Strategies for Effective Paraphrasing. In *Proceedings of COLING-88. Budapest, Hungary, August 22-27*. COLING, 1988.

[36] Meteer, M. *The Spokesman Natural Language Generation System*. BBN Report 7090, BBN Systems and Technologies Corp., Cambridge, MA, 1989.

[37] Miller, H.G., Hershman R.L., and Kelly, R.T. *Performance of a Natural Language Query System in a Simulated Command Control Environment*. Technical Report, Naval Ocean Systems Center, 1978.

[38] Moser, M.G. An Overview of NIKL, the New Implementation of KL-ONE. In Sidner, C. L., et al. (editors), *Research in Knowledge Representation for Natural Language Understanding - Annual Report, 1 September 1982 - 31 August 1983*, pages 7-26. BBN Laboratories Report No. 5421, 1983.

[39] Neff, M. S., and Boguraev, B. K. Dictionaries, Dictionary Grammars, and Dictionary Parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 91-101. 1989.

[40] Obermeier, K.K. Natural-Language Processing. *Byte* :225-232, Dec, 1987.

[41] Ramshaw, L. A. *Research and Development in Natural Language Understanding as Part of the Straegic Computing Program: Annual Technical Report, December 1987 - December 1988.* Technical Report 7047, Bolt Beranek and Newman Inc., 1989.

[42] Ramshaw, L.A. A Metaplan Model for Problem-Solving Discourse. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 35-42. Association for Computational Linguistics, 1989.

[43] Resnik, P. *Access to Multiple Underlying Systems in Janus.* BBN Report 7142, Bolt Beranek and Newman Inc., September, 1989.

[44] Rich, C. Knowledge Representation languages and the Predicate Calculus: How to Have Your Cake and Eat It Too. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 193-196. AAAI, August, 1982.

[45] Shieber, S.M. *An Introduction to Unification-Based Approaches to Grammars.* Center for the Study of Language and Information, Stanford, CA, 1986.

[46] Shieber, S.M., van Noord, G., Moore, R., Pereira, F.C.N. A Semantic Head-Driven Generation Algorithm for Unification-Based Formalisms. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 7-17. Association for Computational Linguistics, 1989.

[47] Thompson, B.H. and Thompson F.B. Rapidly Extendable Natural Language. In *Proceedings of 1978 Annual Conference of the ACM*, pages 173-182. Association for Computing Machinery, Washington, December 4-6, 1978.

[48] Vilain, M. The Restricted Language Architecture of a Hybrid Representation System. In *Proceedings of IJCAI85*, pages 547-551. International Joint Conferences on Artificial Intelligence, Inc., Morgan Kaufmann Publishers, Inc., Los Angeles, CA, August, 1985.

[49] Webber, B.L. *A Formal Approach to Discourse Anaphora.* Technical Report 3761, Bolt, Beranek and Newman, Inc, 1978. Cambridge, Mass.

[50] Webber, B.L. Discourse Model Synthesis: Preliminaries to reference. In A.K. Joshi, B.L. Webber and I.A. Sag (editor). *Elements of Discourse Understanding.* Cambridge University Press, 1981.

[51] Webber, Bonnie L. So What Can We Talk About Now? *Computational Models of Discourse.* MIT Press, 1983, pages 331-372.

[52] Weischedel, R. M. and Sondheimer, N. K. Meta-rules as a Basis for Processing Ill-Formed Input. *American Journal of Computational Linguistics* 9(3-4):161-177, 1983.

[53] Weischedel, R.M. A Personal View of Ill-Formed Input Processing. In *Proceedings of the Workshop on Natural Language Technology Planning.* 1987.

[54] Weischedel, R. M. and Ramshaw, L. A. Reflections on the Knowledge needed to process ill-formed language. *Machine translation.* Cambridge University Press, 1987, Chapter 10.

[55] Weischedel, R., Ayuso, D., Haas, A., Hinrichs, E., Scha. R., Shaked, V., Stallard, D. *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program.* Technical Report, BBN Laboratories, Cambridge, Mass., 1987. Report No. 6522.

[56] Weischedel, R.M. A Hybrid Approach to Representation in the Janus Natural Language Processor. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 193-202. 1989.

[57] Weischedel, R.M., Bobrow, R., Ayuso, D.M., and Ramshaw, L. Portability in the Janus Natural Language Interface. In *Speech and Natural Language*, pages 112-117. Morgan Kaufmann Publishers Inc., San Mateo, CA, 1989.

[58] Weischedel, R., Ayuso, D., Haas, A., Hinrichs, E., Scha, R., Shaked, V., and Stallard, D. *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program, Annual Technical Report December 1985 - December 1986.* BBN Report 6522, Bolt Beranek and Newman, Cambridge, MA, 1987.

[59] Weischedel, R.M. and Ramshaw, L.A. Reflections on the Knowledge Needed to Process Ill-Formed Language. In S. Nirenburg (editor), *Machine Translation: Theoretical and Methodological Issues.* Cambridge University Press, Cambridge, England, 1987.

3. Portability in the Janus Natural Language Interface

Ralph M. Weischedel, Robert J. Bobrow, Damaris Ayuso, Lance Ramshaw[5]

3.1 Introduction: Motivation

Portability is measurable by the person-effort expended to achieve a pre-specified degree of coverage, given an application program. Factoring an NL system into domain-dependent and domain-independent modules is now part of the state of the art; therefore, the challenge in portability is reducing the effort needed to create domain-dependent modules. For us, those are the domain-dependent knowledge bases, e.g., lexical syntax, lexical semantics, domain models, and transformations specific to the target application system.

Our experience in installing our natural language interface as part of DARPA's Fleet Command Center Battle Management Program (FCCBMP) illustrates the kind of portability needed if NL applications (or products) are to become widespread. We demonstrated broad linguistic coverage across 40 fields of a large Oracle database, the Integrated Data Base (IDB), in August 1986. A conclusion was that the state of the art in understanding was adequate. However, the time and cost needed to cover all 400 fields of the IDB in 1986 and the more than 850 fields today would have been prohibitive without a breakthrough in knowledge acquisition and maintenance tools.

We have developed a suite of tools to greatly increase our productivity in porting BBN's Janus NL understanding and generation system to new domains. KREME [1] enables creating, browsing, and maintaining of taxonomic knowledge bases. IRACQ [2] supports learning lexical semantics from examples with only one unknown word. Both of those tools were used in preparing the FCCBMP demonstration in 1986. What was missing was a way to rapidly infer the knowledge bases for the overwhelming majority of words used in accessing fields. If such a tool were available, then one could further bootstrap using IRACQ.

We have developed and used such a tool called KNACQ (for KNowledge ACQuisition). *The efficiency we have experienced results (1) from identifying regularities in expression corresponding to regularities in the domain model and its function and from (2) requiring little information from the user to identify which regularities apply to which domain structures.*

Our long-term goal is to support a seamless interface making simultaneous access to multiple, heterogeneous application systems possible. We have focused thus far on access to expert systems.

---

[5]This paper is a reprint of a paper that appears in **Proceedings of the Speech and Natural Language Workshop**, Morgan Kaufmann Publishers, Inc., San Mateo, CA, February, 1989.

3.2 What KNACQ Does

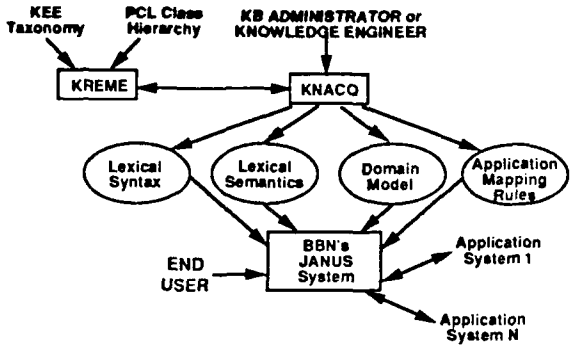

Figure 1: Role of KNACQ

KNACQ assumes that a taxonomic model of the domain exists, such as that typical in many expert systems, and assumes that it is encoded in an axiomatizable subset of KREME [4]. At this point we have built translators for transforming KEE taxonomies and PCL hierarchies into KREME structures.[6] The browsing facilities, graphical views, and consistency checker of KREME are therefore at the disposal of the knowledge base administrator or knowledge engineer when using KNACQ.

Using KREME users may select any concept or role for processing. KNACQ presents the user with a few questions and menus to elicit the English expressions used to refer to that concept or role. There are five cases corresponding to syntactic regularities that reflect functional regularities.

To illustrate the kinds of information that must be acquired consider the examples in Figure 2. To handle these one would have to acquire information on lexical syntax, lexical semantics, and mapping to expert system structure for all words not in the domain-independent dictionary. For purposes of this exposition, assume that the following words, *vessel*, *speed*, *Vinson*, *CROVL*, *C3*, and *deploy* are to be defined. *A vessel has a speed of 20 knots* or *a vessel's speed is 20 knots* would be understood from domain-independent semantic rules regarding *have* and *be*, once lexical information for *vessel* and *speed* is acquired. In acquiring the definitions of *vessel* and *speed*, the system should infer interpretations for phrases such as *the speed of a vessel*, *the vessel's speed*, and *the vessel's speed*.

The vessel speed of Vinson
The vessels with speed above 20 knots
The vessel's speed is 5 knots
Vinson has speed less than 20 knots
Its speed
Which vessels have a CROVL of C3?
Which vessels are deployed C3?

Figure 2

Given the current implementation, the required knowledge for the words *vessel*, *speed*, and *CROVL* is most efficiently acquired using KNACQ; names of instances of classes, such as *Vinson* and *C3* are automatically inferred from instances in the expert system taxonomy; and knowledge about *deploy* and its derivatives would be acquired via IRACQ. That is, *we recommend using IRACQ for the diverse, complex patterns of syntax and semantics arising from verbs by providing examples of the verbs' usage, while using KNACQ for efficient acquisition of the more regular noun phrase information (excluding verb-based constructions).*

[6] Of course, it is not the case that every piece of knowledge statable in KEE taxonomies and PCL hierarchies has a correlate in the axiomatizable subset of KREME. We do not guarantee that there will be English expressions corresponding to anything falling outside of the axiomatizable subset.

3.3 KNACQ Functionality

Five cases are currently handled: one associated with concepts (or frames), two associated with binary relations (or slots), and two for adjectives. In each case, one selects a concept or binary relation (e.g., using the KREME browser) to provide lexicalizations for that domain entity.

3.3.1 Concepts and Classes

The association of English descriptions with concepts is the simplest case. It is fundamental knowledge about unmodified head nouns or frozen nominal compounds from which we can build more powerful examples. KNACQ must acquire one or more phrases for a given class, and their declension, if irregular. For the concept CARRIER of Figure 3, we provide KNACQ with the phrases *carrier* and *aircraft carrier*, which can be treated as a frozen nominal compound. Since both are declined regularly, no further information is required.

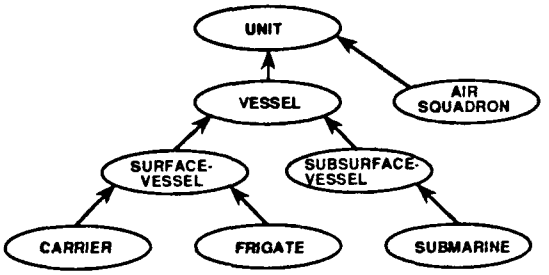

Figure 3: Simple Class Hierarchy

One can provide *surface vessel* for SURFACE-VESSEL in Figure 3, but that would not allow compositions, such as *Count the surface and subsurface vessels.* Rather, one should define *surface* and *subsurface* as non-comparative adjectives (Section 3.4) modifying phrases corresponding to VESSEL in order to define phrases for the concepts SURFACE-VESSEL and SUBSURFACE-VESSEL.

3.3.2 Attributes

Attributes are binary relations on classes that can be phrased as *the <relation> of a <class>*. For instance, suppose CURRENT-SPEED is a binary relation relating vessels to SPEED, a subclass of MEASUREMENT. An attribute treatment is the most appropriate, for *the speed of a vessel* makes perfect sense. KNACQ asks the user for one or more English phrases associated with this functional role; the user response in this case is *speed*. That answer is sufficient to enable the system to understand the kernel noun-phrases listed in Figure 4. Since ONE-D-MEASUREMENT is the range of the relation, the software knows that statistical operations such as average and maximum apply to speed.

KERNEL NOUN PHRASES

the speed of a vessel	*the vessel's speed*
	the vessel speed

COMPOSITIONALLY WITH LEXICAL SEMANTICS, SYNTACTIC RULES, AND SEMANTIC RULES

The vessel speed of Vinson	*Vinson has speed 1*
Carriers with speed 20 knots	*Their average speeds*
The vessel's speed is 5 knots	*Its speed*
Vinson has speed 20 knots	*Their greatest speed*
Which vessels have speeds	
Eisenhower has Vinson's speed	
The carriers with speed above 20 knots	
The vessels with a speed of 20 knots	
Vinson has speed less than 20 knots	
Which vessels have speed above 20 knots	

Figure 4

The lexical information inferred is used compositionally with the syntactic rules, domain independent semantic rules, and other lexical semantic rules. Therefore, the generative capacity of the lexical semantic and syntactic information is linguistically very great, as one would require. A small subset of the examples illustrating this without introducing new domain specific lexical items appears in Figure 4. *It is this compositionality and the domain independent rules that provide the utility of KNACQ.*

3.3.3 Caseframe Rules

Some lexicalizations of roles do not fall within the attribute category. For these, a more general class of regularities is captured by the notion of caseframe rules. Suppose we have a role UNIT-OF, relating CASUALTY-REPORT and MIL-UNIT. KNACQ asks the user which subset of the following six patterns in Figure 5 are appropriate plus the prepositions appropriate.

1. <CASUALTY-REPORT> is <PREP> <MIL-UNIT>
2. <CASUALTY-REPORT> <PREP> <MIL-UNIT>
3. <MIL-UNIT> <CASUALTY-REPORT>
4. <MIL-UNIT> is <PREP> <CASUALTY-REPORT>
5. <MIL-UNIT> <PREP> <CASUALTY-REPORT>
6. <CASUALTY-REPORT> <MIL-UNIT>

Figure 5: Patterns for Caseframe Rules

For this example, the user would select patterns (1), (2), and (3) and select *for*, *on*, and *of* as prepositions. Normally, if pattern (1) is valid, pattern (2) will be as well and vice versa. Similarly, if pattern (4) is valid, pattern (5) will normally be also. As a result, the menu items are coupled by default (selecting (1) automatically selects (2) and vice versa), but this default may be simply overridden by selecting either and then deselecting the other. The most frequent examples where one does not have the coupling of those patterns is the preposition *of*.

3.3.4 Adjectives

3.3.4.1 Gradable Adjectives

Certain attribute roles have ranges that may be compared, e.g., numbers or measurements. Adjectives can be given for these roles; assume *fast* is given by the user for the CURRENT-SPEED role discussed earlier. KNACQ can correctly predict the comparative and superlative forms of *fast*. Suppose x and y are instances of vessel. The next information needed is whether *x is faster than y* means *x's speed is greater than y's speed* or *x's speed is less than y's speed*. Optionally, a threshold t can be given such that *x's speed is greater than t* means *x is fast*. Additionally, one can specify antonyms for *fast*, such as *slow*. The information above would enable understanding the expressions in Figure 6.

Is Frederick faster than every carrier?
Which vessels are slower than 20 knots?
How fast are the carriers?
Show the fastest vessel.
Is Vinson fast?
Is Vinson as fast as Frederick?
How fast is the fastest carrier?

Figure 6: Examples after Defining *Fast*

3.3.4.2 Non-gradable Adjectives

Of the remaining types of adjectives, some correspond to refining a concept to another named concept in the hierarchy. For instance, *surface* and *subsurface* have that property given the network in Figure 3. In such a case, one must indicate the general concept, the refined concept, the adjective, and any synonyms.

Others correspond to an arbitrary restriction on a concept having no explicit

refined concept in the domain model. Though one could add such a refined concept to the hierarchy, we allow the user to state a logical form to define the adjective as a predicate of one argument.

A case that we have not covered in KNACQ is non-gradable adjectives that are predicates of more than one argument. An example in the FCCBMP domain is mission readiness ratings, *M1*, *M2*, *M3*, *M4*, and *M5*. These occur in expressions such as *Enterprise is M2 on anti-air warfare*, where both the vessel and the type of mission are agreements.

3.4 Experience Thus Far

There are several things we have learned even in the early stages of KNACQ's development based on porting Janus to CASES, an expert system in DARPA's Fleet Command Center Battle Management Program (FCCBMP). In this use of KNACQ, the original domain model pertinent to the portion requiring a natural language interface consisted of 189 concepts and 398 roles.

First, no restructuring of that domain model was necessary, nor was any deletion required.

Second, we found it useful to define some additional concepts and roles. Certain subclasses not critical to the expert system were nevertheless lexically significant. In total, only 123 concepts were added: 53 for classes that were treated as strings in the expert system and 70 domain-independent concepts pertaining to time, space, events, commands, etc. Similarly, 28 roles were added: 24 domain-independent roles and 4 domain-specific roles. In addition, some roles were added to represent role chains that are lexically significant directly. For instance, the DISPLACEMENT of the VESSEL-CLASS of a VESSEL is lexicalizable as *the vessel's displacement*. Starting from a given concept, a procedure exists to run through a subhierarchy checking for role chains of length two to ask the user if any of those are significant enough to have lexical forms. For the example network we needed to add only 5 roles for this purpose.

Third, 1093 proper nouns (e.g., ship and port names) were inferred automatically from the instances in the expert system taxonomy.

As a result, the time required to supply lexical syntax and semantics was much less than we had experienced before developing KNACQ. In two days we were able to provide 563 lexical entries (root forms not counting morphological variants) for 103 concepts and 353 roles. Together with the automatically inferred proper nouns, this was approximately 91% of the domain-dependent vocabulary used for the demonstration. That is about 5-10 times more productivity than we had experienced before with manual means.

3.5 Related Work

TEAM [5] is most directly related, having many similar goals, though focussed on data bases rather than expert systems. The novel aspects of KNACQ by contrast with TEAM are (1) accepting an expert system domain model as input (KNACQ) contrasted with the mathematically precise semantics of a relational data base (TEAM) and (2) how little linguistic information is required of the KNACQ user.

A complementary facility is provided in TELI [3] and in LIFER [6]. KNACQ is meant to be used by the (expert system's)

knowledge engineer, who understands the expert system domain model, to define a large portion of the vocabulary, that portion corresponding to simple noun phrase constructions for each concept and role; one uses KNACQ to bootstrap the initially empty domain-dependent lexicon. TELI and LIFER, on the other hand, are meant to let the end user define additional vocabulary in terms of previously defined vocabulary, e.g., *A ship is a vessel*; therefore, those systems assume an extensive vocabulary provided by the system builder. Obviously, providing both kinds of capabilities is highly desirable.

3.6 Conclusions

KNACQ is based on the goal of allowing very rapid, inexpensive definition of a large percentage of the vocabulary necessary in a natural langauge interface to an expert system. It provides the knowledge engineer with the facilities to browse his/her taxonomic knowledge base, and to state head nouns, nominal compounds, and their non-clausal modifiers for referring to the concepts and roles in the knowledge base. Given that, KNACQ infers the necessary lexical syntactic and lexical semantic knowledge. Furthermore, if appropriate instances in the expert system knowledge base already have names, KNACQ will add proper nouns for those instances to the lexicon.

KNACQ does not cover the inference of complex constructions typical of verbs and their nominalizations. IRACQ [2] allows a user to enter examples of usage for acquiring lexical syntax and semantics for complex constructions.

Our experience thus far is that KNACQ has achieved our goals of dramatically reducing the time it takes to define the vocabulary for an expert system interface. It appears to have increased our own productivity several fold. (However, KNACQ has not yet been provided to a knowledge engineer with no knowledge of computational linguistics.)

We believe that the problem of linguistic knowledge acquisition is critical not just as a practical issue regarding widespread availability of natural language interfaces. As our science, technology, and systems become more and more mature, the ante to show progress could involve more and more effort in filling domain-specific knowledge bases. The less effort spent on such knowledge bases, the more effort can be devoted to unsolved problems.

References

[1] Abrett, G. and Burstein, M. The KREME Knowledge Editing Environment. *Int. J. Man-Machine Studies* 27:103-126, 1987.

[2] Ayuso, D.M., Shaked, V., and Weischedel, R.M. An Environment for Acquiring Semantic Information. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 32-40. ACL, 1987.

[3] Ballard, B. and Stumberger, D. Semantic Acquisition in TELI: A Transportable, User-Customized Natural Language Processor. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 20-29. ACL, June, 1986.

[4] Brachman, R.J. and Schmolze, J.G. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9(2), April, 1985.

[5] Grosz, B., Appelt, D. E., Martin, P., and Pereira, F. TEAM: An Experiment in the Design of Transportable Natural Language Interfaces. *Artificial Intelligence* , 1985.

[6] Hendrix, G., et al. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2):105-147, 1978.

4. Discourse Entities in Janus

Damaris M. Ayuso[7]

Abstract

This paper addresses issues that arose in applying the model for discourse entity (DE) generation in B. Webber's work (1978, 1983) to an interactive multi-modal interface. Her treatment was extended in 4 areas: (1) the notion of context dependence of DEs was formalized in an intensional logic, (2) the treatment of DEs for indefinite NPs was modified to use skolem functions, (3) the treatment of dependent quantifiers was generalized, and (4) DEs originating from non-linguistic sources, such as pointing actions, were taken into account. The discourse entities are used in intra- and extra-sentential pronoun resolution in BBN Janus.

4.1 Introduction

Discourse entities (DEs) are descriptions of objects, groups of objects, events, etc. from the real world or from hypothesized or possible worlds that are evoked in a discourse. Any communicative act, be it spoken, written, gestured, or system-initiated, can give rise to DEs. As a discourse progresses, an adequate discourse model must represent the relevant entities, and the relationships between them [4]. A speaker may then felicitously refer anaphorically to an object (subject to focusing or centering constraints (Grosz et al., 1983, Sidner 1981, 1983, Brennan et al. 1987)) if there is an existing DE representing it, or if a corresponding DE may be directly inferred from an existing DE. For example, the utterance "Every senior in Milford High School has a car" gives rise to at least 3 entities, describable in English as "the seniors in Milford High School", "Milford High School", and "the set of cars each of which is owned by some senior in Milford High School". These entities may then be accessed by the following next utterances, respectively:

"They graduate in June."
"It's a good school."
"They completely fill the parking lot."

Webber (1978, 1983) addressed the question of determining what discourse entities are introduced by a text. She defined rules which produce "initial descriptions" (IDs) of new entities stemming from noun phrases, given a meaning representation of a text. An ID is a logical expression that denotes the corresponding object and uses only information from the text's meaning representation. The declarative nature of Webber's rules and the fact that they relied solely on the structure of the meaning representation, made her approach well suited for implementation.

---

[7]This paper is reprinted from the *Proceedings of the 27th Annual Meeting of the Association for Compuational Linguistics*, 26-29 June 1989, University of British Columbia, Vancouver, BC Canada. Requests for copies should be addressed to:

Dr. Donald E. Walker (ACL)
Bell Communications Research
435 South Street MRE 2A379
Morristown, NJ 07960, USA

The present work recasts her rules in Janus's intensional logic framework (described in section 4.2). Two goals guided our approach: (1) that our DE representations be semantically clear and correct according to the formal definitions of our language, and (2) that these representations be amenable to the processing required in an interactive environment such as ours, where each reference needs to be fully resolved against the current context.

In the following sections, we first present the representational requirements for this approach, and introduce our logical language (section 4.2).

Then we discuss issues that arose in trying to formalize the logical representation of DEs with respect to (1) the context dependence of their denotations, and (2) the indeterminacy of denotation that arises with indefinite NPs. For context dependence, we use an intensional logic expression indexed by time and world indices (discussed in section 4.3). This required us to extend Webber's rules to detect modal and other index-binding contexts. In representing DEs for indefinites (appearing as existential formulae in our meaning representation), we replaced Webber's EVOKE predicate with skolem constants for the independent case, where it does not contain a variable bound by a higher FORALL quantifier (section 4.4), and do not use EVOKE at all in the dependent case.

In section 4.5 we introduce a generalized version of the rules for generating DEs for dependent quantifiers stemming from indefinite and definite NPs which overcomes some difficulties in capturing dependencies between discourse entities.

In our multi-modal interface environment, it is important to represent the information on the computer screen as part of the discourse context, and allow references to screen entities that are not explicitly introduced via the text input. Section 4.6 briefly discusses some of these issues and shows how pointing actions are handled in Janus by generating appropriate discourse entities that are then used like other DEs.

Finally, section 4.7 concludes and presents plans for future work.

This is, to our knowledge, the first implementation of Webber's DE generation ideas. We designed the algorithms and structures necessary to generate discourse entities from our logical representation of the meaning of utterances, and from pointing gestures, and currently use them in Janus's [22, 1] pronoun resolution component, which applies centering techniques (Grosz et al., 1983, Sidner 1981, 1983, Brennan et al. 1987) to track and constrain references. Janus has been demonstrated in the Navy domain for DARPA's Fleet Command Center Battle Management Program (FCCBMP), and in the Army domain for the *Air Land Battle* Management Program (ALBM).

4.2 Meaning Representation for DE Generation

Webber found that appropriate discourse entities could be generated from the meaning representation of a sentence by applying rules to the representation that are strictly structural in nature, as long as the representation reflects certain crucial aspects of the sentence. This has the attractive feature that any syntactic formalism may be used if an appropriate semantic representation is produced. Some of the requirements (described in (Webber 1978, 1983)) on the representation are: (1) it must distinguish between definite and indefinite NPs and between singular and plural NPs, (2) it must specify quantifier scope, (3) it must distinguish between distributive and collective

readings, (4) it must have resolved elided verb phrases, and (5) it must reflect the modifier structure of the NPs (e.g., via restricted quantification). An important implied constraint is that the representation must show one recognizable construct (a quantifier, for example) per DE-invoking noun phrase. These constructs are what trigger the DE generation rules.

Insofar as a semantic representation reflects all of the above in its structure, structural rules will suffice for generating appropriate DEs, but otherwise information from syntax or other sources may be necessary. There is a trade-off between using a level of representation that shows the required distinctions, and the need to stay relatively close to the English structure in order to only generate DEs that are justified by the text. For example, in Janus, in addition to quantifiers from NPs, the semantic representation has quantifiers for verbs (events), and possibly extra quantifiers introduced in representing deeper meaning or by the collective/distributive processing. Therefore, we check the syntactic source of the quantifiers to ensure that we only generate entities for quantifiers that arose from NPs (using the bound variable as an index into the parse tree).

Other than the caveat just discussed, the Janus meaning representation language WML (for World Model Language) [6] meets all the other constraints for DE generation. WML is a higher-order intensional language that is based on a synthesis between the kind of language used in PHLIQA [12] and Montague's Intensional Logic [11]. A newer version of WML [16] is used in the BBN Spoken Language System [2]. The intensionality of WML makes it more powerful than the sample language Webber used in developing her structural rules.

The scoping expressions in WML have a sort field (which restricts the range of the variable) and have the form:

```
(B x S (P x))
```

where B is a quantifier such as FORALL or EXISTS, a term-forming operator like IOTA or SET, or the lambda abstraction operator LAMBDA. S is the sort, a set-denoting expression of arbitrary complexity specifying the range of x, and (P x) is a predication in terms of x. The formal semantics of WML assigns a type to each well-formed expression which is a function of the types of its parts. If expression E has type T, the denotation of E, given a model M and a time t and world w, is a member of the set which is T's domain. One use of types in our system is for enforcing selectional restrictions. The formation rules of WML, its type system, and its recursive denotation definition provide a formal syntax and semantics for WML.

4.3 Context Dependence of Discourse Entities

A formal semantics was assumed though not given for the sample logical language used by Webber. The initial descriptions (IDs) of DEs produced by her rules were stated in this language too, and thus are meant to denote the object the DE represents. For example, the rule which applies to the representation for independent definite NPs assigns to the resulting DE an ID which is the representation itself:

```
(ι x S (P x)) =>
      ID: (ι x S (P x))
```

where ι is Russell's iota operator. Thus, the ID for "the cat" in "I saw the cat" is (ι x cats T). (Since the body of the ι in this example has no additional predication on x, it is merely T, for TRUE.) However, because IDs are solely drawn from the meaning representation of the isolated text, they may not suffice to denote a unique object. Connection to prior discourse knowledge or

information from further discourse may be necessary to establish a unique referent, or determining the referent may not even be necessary. For example, the ID for "the cat" would need to be evaluated in a context where there is only one salient cat in order to obtain a denotation.

Our system's representation of a DE is a structure containing several fields. The "logical-form" field contains a WML expression which denotes the object the DE describes (this corresponds roughly to Webber's ID). Given that WML is intensional, we are able to explicitly represent context dependence by having the logical form include an intensional core, plus tense, time, and world information (which includes discourse context) that grounds the intension so that it may be evaluated. For example, the logical form for the DE corresponding to "the cat" in our system is

```
((INTENSION (IOTA x cats T))
 time world)
```

where *time*, if unfilled, defaults to the present, and *world* defaults to the real world and current discourse state. The semantics of our IOTA operator makes it denotationless if there is not exactly one salient object that fits the description in the context, else its denotation is that unique object. In our interactive system each reference needs to be fully resolved to be used successfully. If unknown information is necessary to obtain a *unique* denotation for a IOTA term, a simple clarification dialogue should ensue. (Clarification is not implemented yet, currently the set of all values fitting the IOTA is used.)

An example using the time index is the noun phrase "the ships that were combat ready on 12/1/88", which would generate a DE with logical form:

```
((INTENSION
  (PAST (INTENSION
   (IOTA x (SETS ships)
      (COMBAT-READY x)))))
 12/1/88 world)
```

Representing this time index in the logical form is crucial, since a later reference to it, made in a different time context must still denote the original object. For example, "Are they deployed?" must have "they" refer to the ships that were combat ready on 12/1/88, not at the time of the latter utterance.

In order to derive the proper time and world context for the discourse entities, we added structural rules that recognize intensional and index-binding logical contexts. Our DE generation algorithm uses these rules to gather the necessary information as it recurses into the logical representation (applying rules as it goes) so that when a regular rule fires on a language construct, the appropriate outer-scoping time/world bindings will get used for the generated DEs.

It should be noted that, as the discussion above suggests, a definite NP always gives rise to a new discourse entity in our system. If it is determined to be anaphoric, then a pointer to the DE it co-refers with (when found) will be added to its "refers-to" field, indicating they both denote the same object.

4.4 DEs for Independent Indefinite NPs

In Webber's work, the initial description (ID) for a DE stemming from an independent existential (i.e., with no dependencies on an outer FORALL quantifier), contained an EVOKE predicate. "I saw a cat":

```
(EXISTS x cats (saw I x))
```

would generate a DE with ID:

```
(ι x cats
   (& (saw I x)
      (EVOKE Sent x)))
```

"The cat I saw that was evoked by sentence Sent", where Sent is the parsed clause for

"I saw a cat". The purpose of EVOKE was to make clear that although more than one cat may have been seen, the "a" picks out one in particular (which one we do not know except that it is the one mentioned in the utterance), and this is the cat which makes the EVOKE true. Any subsequent reference then picks out the same cat because it will access this DE. The semantics of the EVOKE predicate and the type of the S argument (which is syntactic in nature) were unclear, so we looked for a different formulation with better understood semantics.

Predicate logic already provides us with a mechanism for selecting arbitrary individuals from the domain via skolem functions (used as a mechanism for removing existentials from a formula while preserving satisfiability). Skolem functions have been used in computational linguistics to indicate quantifier scope, for example [17]. Following a suggestion by R. Scha, we use skolem functions in the logical form of the DE for the "indefinite individuals" introduced by independent existentials [13]. For clarity and consistency with the rest of the language, we use a *sorted* skolem form, where the range of the function is specified. Since we use this for representing existentials that are independent, the function has no arguments and is thus equivalent to a sorted constant whose denotation is undetermined when introduced. (In this sense it is consistent with Karttunen's (1976) and Kamp's (1984) view of the indefinite's role as a referential constant, but unlike Kamp, here the sentence's meaning representation is separate from the representation of the evoked entity.)

Thus we introduced a new operator to WML named SKOLEM, for expressions of the form (SKOLEM *n* <sort>), where *n* is an integer that gets incremented for each new skolem created, as a way of naming the skolem function. For the example above, the core logical form (stripping the outer intension and indices) for the DE of "a cat" would be:

```
(SKOLEM 1
   (SET x cats (saw I x)))
```

denoting a particular cat from the set of all the cats I saw. The type of a SKOLEM expression is well-defined and is given by the following type rule:

```
TYPEOF (SKOLEM INTEGERS
                (SETS a))
       = a
```

where INTEGERS is the type for integers, and (SETS *a*) is the type of sets whose members have type *a*. This type rule says that when the first argument of SKOLEM is of type INTEGER, and the second is a set with elements of type *a*, then the type of the SKOLEM expression is *a*. Therefore, the type of the above example is cats. The explicit connection to the originating sentence which the EVOKE predicate provided is found in our scheme outside of the logical representation by having a pointer in the DE's structure to the parse tree NP constituent, and to the structure representing the communicative act performed by the utterance (in the fields "corresponding-constituent" and "originating-communicative-act", respectively). These connections are used by the pronoun-resolution algorithms which make use of syntactic information.

Does the denotation of a skolem constant ever get determined? In narrative, and even in conversation, identifying the individual referred to by the indefinite NP frequently doesn't occur. However, in our interactive system, each reference must be fully resolved. When the evaluation component of Janus determines a successful value to use for the existential in the text's logical form, the appropriate function denotation for SKOLEM *n* gets defined, and the "extension" field is set for the discourse entity.

Note that many interesting issues come up in the treatment of reference to these indefinite entities in a real system. For ex-

ample, cooperative responses by the system introduce new entities that must be taken into account. If the user asks "Is there a carrier within 50 miles of Hawaii?", a cooperative "There are two: Constellation and Kennedy" (as opposed to just "Yes") must add those two carriers as entities, which now overshadow the singular skolem entity for "a carrier within 50 miles of Hawaii". On the other hand, a "No" answer should block any further reference to the carrier skolem, since its denotation is null, while still allowing a reference to a class entity derived from it, as in "Is there *one* near San Diego?" where *one* refers to the class *carriers*.

The treatment presented works for straightforward cases of independent indefinites. Trickier cases like donkey sentences [8, 19] and interactions with negation have not yet been addressed.

4.5 Dependent NPs

4.5.1 Dependent Indefinite NPs

Our work uncovered a need for modifications in Webber's structural rules for quantifiers from indefinite and definite NPs which have dependencies on variables bound directly or indirectly by an outer FORALL quantifier. In this section we address the case of dependent existentials arising from indefinite NPs. We first argue that the predicate EVOKE is not needed in this context. Then we point out the need for generalizing the rule to take into account not just FORALL, but all scoping operators that intervene between the outer FORALL and the inner EXISTS. Finally, we show that the dependencies between discourse entities must be explicitly maintained in the logical forms of newly created DEs that depend on them.

Webber's rules are designed to apply from the outermost quantifier in; each time a rule is applied the remaining logical form is modified to be in terms of the just created DE. For example, "Every boy saw a girl he knows" has logical form (for the bound pronoun reading):

```
(FORALL x boys
 (EXISTS y (SET y' girls
             (knows x y'))
    (saw x y)))
```

The first step is to apply the rule for an independent universal quantifier:

```
R0:
 (FORALL x S (P x)) =>  de: S]
```

This application yields the entity for "the set of all boys"

```
DE1: boys
```

and we rewrite the logical form to be:

```
(FORALL x DE1
  (EXISTS y (SET y' girls
              (knows x y'))
     (saw x y)))
```

The steps shown so far are consistent with both Webber's and our approach. Now we want to apply the general rule for existentials within the body of a distributive, in order to generate an entity for the relevant set of girls. Webber uses Rule 3 in [20] (here corrected to position the existential's sort S inside the scope of the outer quantifiers in the generated DE):

```
R3: (FORALL y1...yk
       (EXISTS x S (P x))) =>
 de: (SET x things
       (EXISTS y1...yk
         (& (member x S) (P x)
            (EVOKE Sent x))))
```

where FORALL $y_1 \ldots y_k$ is shorthand for FORALL y_1 de_1 (...(FORALL y_k de_k, analogously for EXISTS, and S or P depends directly or indirectly on $y_1 \ldots y_k$.

Now the first DE we want to generate with this rule is for "the set of girls, each of which is known by some boy in DE_1, and was seen by him". Does each girl in the set

also have to satisfy an EVOKE predicate? It seems that any future reference back to the set formed by the existential seeks to obtain *all* items fitting the description, not some subset constrained by EVOKE. For example, if the example above is followed by "the girls tried to hide", taking "the girls" anaphorically, one wants *all* the girls seen by some boy in DE_1 that knows them, no less. Our core logical representation for the set of girls is thus:

```
DE2: (SET y girls
       (EXISTS x DE1
        (& (knows x y)
           (saw x y))))
```

So the modified rule used in producing DE_2 is:

```
R3': (FORALL y1...yk
        (EXISTS x S (P x))) =>
   de: (SET x St
         (EXISTS y1...yk
           (& (member x S)
              (P x))))
```

where EVOKE has been removed, and the DE's sort field is S_t for the "root type" of S, which is the type of the members of S, in order to appropriately constrain the DE's sort (instead of leaving it as the unconstrained "things").

A second change that needs to be made is to generalize the left hand side of the rule so that the scoping expressions outscoping the inner EXISTS in the pattern also be allowed to include other scoping operators, such as EXISTS and IOTA. As long as the outermost quantifier is a FORALL, any other dependent scoping expression within it will generate a set-denoting DE and will behave as a distributive environment as far as any more deeply embedded expressions are concerned. In other words, the distributiveness chains along the dependent quantifiers. To see this, consider the more embedded example "Every boy gave a girl he knew a peach she wanted", where there is an intervening existential between the outer FORALL and innermost EXISTS. The core logical form for this sentence is:

```
(FORALL x boys
 (EXISTS y (SET y' girls
             (knows x y'))
  (EXISTS z (SET z' peaches
              (wants y z'))
    (gave x y z))))
```

DE_1 would be as above. Using rule R3' DE_2 becomes:

```
DE2:
 (SET y girls
  (EXISTS x DE1
   (& (knows x y)
     (EXISTS z (SET z' peaches
                 (wants y z'))
       (gave x y z)))))
```

"The set of girls, each of which is known by some boy in DE_1, and got a peach she wanted from that boy." Now the peach quantifier should generate a set DE in terms of DE_1 and DE_2. Applying R3' gives us:

```
DE3:
 (SET z peaches
   (EXISTS x DE1
     (EXISTS y DE2
       (& (wants y z)
          (gave x y z)))))
```

"The set of peaches z such that there is a girl in DE_2 (who is known by some boy in DE_1, and who got some peach she wanted from the boy), who wants z, and who got it from some boy in DE_1".

Now a third and final problem becomes apparent: for the general case of arbitrary embedding of dependent quantifiers we generate a DE (e.g., DE_3) dependent on other DEs from the outer quantifiers, but the *dependencies between those DEs* (e.g., DE_1 and DE_2) are not maintained. This is counter-intuitive, and also leads to an underspecified set DE. In the peaches example above, envision the situation where a boy b_1 gave out two peaches p_1 and p_2 : one to a girl g_1 he knew, and one to a girl g_2 he *didn't* know, who also got a peach p_3 from another boy b_2 who *did* know her. These are the facts of interest in this scenario:

```
1. (& (gave b1 g1 p1)  (know b1 g1)
      (wants g1 p1))
2. (& (gave b1 g2 p2)
      (NOT (know b1 g2))
      (wants g2 p2))
3. (& (gave b2 g2 p3)  (know b2 g2)
      (wants g2 p3))
```

Since b_1 and b_2 are in DE_1 (due to facts 1 and 3), and g_2 is in DE_2 (due to fact 3), then p_2 is in DE_3 (due to fact 2 and according to the DE_3 logical form above). But p_2 *should not* be in DE_3, since p_2 was NOT given to a girl by a boy *she knew*. The set of peaches obtained for DF_3 is too large. The problem would not arise if in the DE_3 logical form, the variables ranging over DE_2 were appropriately connected to DE_1 using the dependent restriction present in the original formula *(knows x y)*. A correct DE_3 is:

```
DE3:
 (SET z peaches
   (EXISTS x DE1
     (EXISTS y (SET y' DE2
                  (knows x y'))
       (& (wants y z)
          (gave x y z)))))
```

To be able to do this, the rule-application algorithm must be modified to include the restriction information (for dependent restrictions) when the formula gets rewritten in terms of a newly created DE. Therefore the final generalized rule, which includes other scoping operators and works on properly connected DEs is as follows:

```
R3'':
 (FORALL v1 S1
   (Q2 v2 S2 ... Qn vn Sn
     (EXISTS x S (P x)))) =>
 de: (SET x St
       (EXISTS v1 S1 ...vn Sn
         (& (member x S)
            (P x))))
```

where S or P depend directly or indirectly on $v_1 \ldots v_n$, Q_i may be FORALL, EXISTS, or IOTA, and the scoping operators outside the inner EXISTS have already been processed by any appropriate rules that have replaced their original sorts by the S_is, which are in terms of generated DEs and explicitly show any DE dependencies. The right hand side is as before, with existentials picking out elements from each outer quantifier.

4.5.2 Dependent Definite NPs

Some of the problems described in the previous section also arise for the rule to handle dependent definite NPs. Definite NPs are treated as IOTA terms in WML. (Webber's logical language in [18] used a similar ι. The treatment was later changed [20] to use the definite existential quantifier "Exists!", but this difference is not relevant for the following.) Replacing IOTA for ι in Webber's (1978) rule 5:

```
R5: (FORALL y1...yk
      (P (IOTA x S (R x)))) =>
 de: (SET z things
       (EXISTS y1...yk
         (= z
            (IOTA x S (R x)))))
```

where $y_1 \ldots y_k$ are universal quantifiers over DEs as in R3 above, and S or R depend directly or indirectly on $y_1 \ldots y_k$.

The second and third extensions discussed in the previous section are needed here too: generalizing the quantifiers that outscope the inner existential, and keeping the dependencies among the DEs explicit to avoid under-specified sets. An example of an under-specified set arises when the dependent IOTA depends jointly on more than one outer variable; for example, in "Every boy gave a girl he knew *the* peach *they* selected", each peach depends on the selection by a boy and a girl together. Take a scenario analogous to that in the previous section, with the facts now as follows (replacing "selected" for "wants"):

```
1. (& (gave b1 g1 p1) (know b1 g1)
      (selected
        (SETOF b1 g1) p1))
2. (& (gave b1 g2 p2)
      (NOT (know b1 g2))
      (selected
        (SETOF b1 g2) p2))
3. (& (gave b2 g2 p3) (know b2 g2)
      (selected
        (SETOF b2 g2) p3))
```

By an analogous argument as before, using R5, the set of peaches will incorrectly contain p_2 , given by a boy to a girl who selected it with him, but whom he did not know. The modified rule is analogous to R3'' in the previous section:

```
R5' :
 (FORALL v1 S1
   (Q2 v2 S2 ... Qn vn Sn
     (P (IOTA x S (R x))))) =>
 de: (SET z St
      (EXISTS v1 S1 ...vn Sn
       (= z
          (IOTA x S (R x))))))
```

Note that this problem of underspecified sets does not arise when the dependency inside the IOTA is on one variable, because the definite "the" forces a one-to-one mapping from the possible assignments of the single outer variable represented in the IOTA to the IOTA denotations. If we use the example, ''Every boy gave a girl he knew *the* peach she wanted'', with logical form:

```
(FORALL x boys
 (EXISTS y (SET y' girls
                (knows x y'))
   (gave x y (IOTA z peaches
                (wants y z)))))
```

there is such a mapping between the set of girls in the appropriate DE_2 (those who got the peach they wanted from a boy they knew) and the peaches in DE_3 obtained via R5' (the peaches that some girl in DE_2 wanted). Each girl wants exactly one peach, so facts 2 and 3, where the same girl receives two different peaches, cannot occur. So the definite ensures that no scenario can be constructed containing extra items, as long as there is only one outer variable in the inner iota. However in the joint dependency example above using "selected", the one-to-one mapping is between boy-girl *pairs* and peaches, so the relationship between the boys and the girls becomes an integral part of determining the correct DE_3.

4.6 Non-Linguistic Discourse Entities

In a dialogue between persons, references can be made not only to linguistically-introduced objects, but also to objects (or events, etc.) that become salient in the environment through some non-linguistic means. For example, a loud noise may prompt a question ''What was *that*?'', or one may look at or point to an object and refer to it, ''What's wrong with *it*?'' It seems an attention-drawing event normally precedes such a reference.

In the Janus human-computer environment, non-linguistic attention-drawing mechanisms that we have identified so far include pointing actions by the user, and highlighting (by the system) of changes on the screen as a response to a request (or for other reasons). The appearance of answers to questions also draws the user's attention. We incorporated these into generalized notion of a ''communicative act'' which may be linguistic in nature (English input or generated English output), a pointing gesture by the user, or some other system-initiated action. Any communicative act may give rise to DEs and affect the focused entities in the discourse.

We have implemented procedures to handle pointing actions by generating discourse entities which are then used in the pronoun resolution component uniformly with the others. For example, after the re-

quest "Show me the C1 carriers in the Indian Ocean" the system will display icons on the color monitor representing the carriers. The user can then say "Which of them are within 200 miles of it? <point with mouse to Kennedy>". Before the sentence gets processed, a discourse entity with the logical form
(IOTA x carriers (nameof x "Kennedy"))
will be created and added to the list of entities currently in focus (the "forward looking centers" of the last linguistic act); the DE's "originating-communicative-act' field will point to a newly created "pointing" communicative act. Since "them" and "it" have different number requirements, there is no ambiguity and the anaphor resolution module resolves "them" to the DE corresponding to "the C1 carriers in the Indian Ocean" and "it" to the DE for Kennedy. We are currently working on having system-initiated actions also generate entities.

4.7 Conclusions and Further Work

Webber's general approach to discourse entity generation from a logical representation proved very useful in our efforts. We were able to recast her basic ideas in our logical framework, and currently use the generated DEs extensively.

The fact that the generation of DEs is done via structural rules operating on a semantic representation provided a degree of modularity that allowed our pronoun resolution component to work *automatically* when we combined a new syntactic component with our semantic and discourse component (replacing an ATN by a unification grammar, in an independently motivated experiment). We are currently starting to port the DE generation component to the BBN Spoken Language System [2], and plan to integrate it with the intra-sentential mechanisms in [7]. The fact that entity representations are mostly semantic in nature, not syntactic, also facilitated the addition and use of non-linguistic entities in a *uniform way*.

There are several areas that we would like to study to extend our current treatment. We want to address the interactions between centering phenomena and non-linguistic events that affect discourse focus, such as changing contexts via a menu selection in an expert system.

Our paraphrasing component [10] already uses the discourse entities to a limited extent. One area of future work is to have the language generator make more extensive use of them, so it can smoothly refer to focused objects.

Finally, although quantified expressions are already generated in Janus for events implicit in many verbs, they are not being used for DEs. We would like to address the problem of event reference and its interaction with temporal information, using ideas such as those in [21] and in the special issue of *Computational Linguistics* on tense and aspect (Vol. 14, Number 2 June 1988).

4.8 Acknowledgments

The author would like to thank Dave Stallard for invaluable discussions during the writing of this paper. Thanks also to Remko Scha, Lance Ramshaw, Ralph Weischedel, and Candy Sidner.

References

[1] . *A Guide to IRUS-II Application Development in the FCCBMP*. BBN Report 6859, Bolt Beranek and Newman Inc., Cambridge, MA, 1988.

[2] Boisen, S., Chow Y., Haas, A, Ingria, R., Roucos, S., Scha, R., Stallard, D., and Vilain, M. *Integration of Speech and Natural Language: Final Report*. BBN Report 6991, BBN Systems and Technologies Corp., 1989.

[3] Brennan, Susan E., Friedman, Marilyn W., and Pollard, Carl J. A Centering Approach to Pronouns. In *Proceedings of the 25th Annual Meeting of the ACL*, pages 155-162. ACL, 1987.

[4] Grosz, Barbara J., and Sidner, Candace L. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics* 12(3):175-204, 1986.

[5] Grosz, Barbara J., Joshi, Aravind K., Weinstein, Scott. Providing a Unified Account of Definite Noun Phrases in Discourse. In *Proceedings of the 21st Annual Meeting of the ACL*, pages 44-50. ACL, Cambridge, MA, 1983.

[6] Hinrichs, E.W., Ayuso, D.M., and Scha, R. The Syntax and Semantics of the JANUS Semantic Interpretation Language. In *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program, Annual Technical Report December 1985 - December 1986*, pages 27-31. BBN Laboratories, Report No. 6522, 1987.

[7] Ingria, Robert J.P., and Stallard, David. A Computational Mechanism for Pronominal Reference. In *Proceedings of the 27th Annual Meeting of the ACL*. ACL, 1989.

[8] Kamp, Hans. A Theory of Truth and Semantic Representation. In J. Groenendijk, T.M.V. Janssen, and M. Stokhof (editors), *Truth, Interpretation and Information, Selected Papers from the Third Amsterdam Colloquium*, pages 1-41. Foris Publications, Dordrecht, 1984.

[9] Karttunen, Lauri. Discourse Referents. In J. D. McCawley (editor), *Syntax and Semantics, Volume 7*, pages 363-385. Academic Press, New York, 1976.

[10] Meteer, Marie and Shaked, Varda. Strategies for Effective Paraphrasing. In *Proceedings of COLING-88. Budapest, Hungary, August 22-27*. COLING, 1988.

[11] Montague, Richard. The Proper Treatment of Quantification in Ordinary English. In J. Hintikka, J. Moravcsik and P. Suppes (editors), *Approaches to Natural Language*, pages 221-242. Reidel, Dordrecht, 1973.

[12] Scha, Remko J.H. Semantic Types in PHLIQA1. In *Coling 76 Preprints*. Ottawa, Canada, 1976.

[13] Scha, Remko J.H., Bruce, Bertram C., and Polanyi, Livia. Discourse Understanding. In *Encyclopedia of Artificial Intelligence*, pages 233-245. John Wiley & Sons, Inc., 1987.

[14] Sidner, Candace L. Focusing for the Interpretation of Pronouns. *American Journal of Computational Linguistics* 7(4):217-231, 1981.

[15] Sidner, Candace L. Focusing in the Comprehension of Definite Anaphora. In M. Brady and R. C. Berwick (editors), *Computational Models of Discourse*, chapter 5, pages 267-330. MIT Press, Cambridge, MA, 1983.

[16] Stallard, David G. A Manual for the Logical Language of the BBN Spoken Language System. 1988.Unpublished.

[17] Kurt VanLehn. *Determining the Scope of English Quantifiers*. Technical Report 483, MIT Artificial Intelligence Laboratory, 1978.

[18] Webber, Bonnie L. *A Formal Approach to Discourse Anaphora*. BBN Report 3761, Bolt Beranek and Newman, Cambridge, MA, 1978.

[19] Webber, Bonnie L. Discourse Model Synthesis: Preliminaries to Reference. *Elements of Discourse Understanding*. Cambridge University Press, 1981, pages 283-299.

[20] Webber, Bonnie L. So What Can We Talk About Now? *Computational Models of Discourse*. MIT Press, 1983, pages 331-372.

[21] Webber, Bonnie L. Discourse Deixis: Reference to Discourse Segments. In *Proceedings of the 26th Annual Meeting of the ACL*, pages 113-122. ACL, 1988.

[22] Weischedel, R., Ayuso, D., Haas, A., Hinrichs, E., Scha, R., Shaked, V., and Stallard, D. *Research and Development in Natural Language Understanding as Part of the Strategic Computing Program, Annual Technical Report December 1985 - December 1986*. BBN Report 6522, Bolt Beranek and Newman, Cambridge, MA, 1987.

5. A Metaplan Model for Problem-Solving Discourse

Lance A. Ramshaw[8]

ABSTRACT

The structure of problem-solving discourse in the expert advising setting can be modeled by adding a layer of metaplans to a plan-based model of the task domain. Classes of metaplans are introduced to model both the agent's gradual refinement and instantiation of a domain plan for a task and the space of possible queries about preconditions or fillers for open variable slots that can be motivated by the exploration of particular classes of domain plans. This metaplan structure can be used to track an agent's problem-solving progress and to predict at each point likely follow-on queries based on related domain plans. The model is implemented in the Pragma system where it is used to suggest corrections for ill-formed input.

5.1 Introduction

Significant progress has been achieved recently in natural language (NL) understanding systems through the use of plan recognition and "plan tracking" schemes that maintain models of the agent's domain plans and goals. Such systems have been used for recognizing discourse structure, processing anaphora, providing cooperative responses, and interpreting intersentential ellipsis. However, a model of the discourse context must capture more than just the plan structure of the problem domain. Each discourse setting, whether argument, narrative, cooperative planning, or the like, involves a level of organization more abstract than that of domain plans, a level with its own structures and typical strategies. Enriching the domain plan model with a model of the agent's plans and strategies on this more abstract level can add significant power to an NL system. This paper presents an approach to pragmatic modeling in which metaplans are used to model that level of discourse structure for problem-solving discourse of the sort arising in NL interfaces to expert systems or databases.

The discourse setting modeled by metaplans in this work is expert-assisted problem-solving. Note that the agent's current task in this context is creating a plan for achieving the domain goal, rather than executing that plan. In problem-solving discourse, the agent poses queries to the expert to gather information in order to select a plan from among the various possible plans. Meanwhile, in order to respond to the queries cooperatively, the expert must maintain a model of the plan being considered by the agent. Thus the expert is in the position of deducing from the queries that are the

[8]This paper is a reprint from the *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, 10-12 April 1989, University of Manchester, Manchester, England. Requests for copies should be addressed to:

Dr. Donald E. Walker (ACL)
Bell Communications Research
435 South Street MRE 2A379
Morristown, NJ 07960, USA

agent's observable behavior which possible plans the agent is currently considering. The metaplans presented here model both the agent's plan-building choices refining the plan and instantiating its variables and also the possible queries that the agent may use to gain the information needed to make those choices. This unified model in a single formalism of the connection between the agent's plan-building choices and the queries motivated thereby allows for more precise and efficient prediction from the queries observed of the underlying plan-building choices. The model can be used for plan tracking by searching outward each time from the previous context in a tree of metaplans to explore the space of possible plan-building moves and related queries, looking for a predicted query that matches the agent's next utterance. Thus the examples will be presented in terms of the required search paths from the previous context to find a node that matches the context of the succeeding query.

This metaplan model is discussed in two parts, with Section 2 covering the plan-building class of metaplans, which model the agent's addition of new branches to the domain plan tree and instantiation of variables, while Section 3 presents examples of plan feasibility and slot data query metaplans, which model the agent's strategies for gathering information to use in plan-building. Section 4 then compares this modeling approach to other plan-based styles of discourse modeling, Section 5 discusses applications for the approach and the current implementation, and Section 6 points out other classes of metaplans that could be used to broaden the coverage of the model and other areas for further work.

5.2 Plan Building Metaplans

In this approach, the plan-building metaplans discussed in this section model those portions of problem-solving behavior that explore the different possible refinements of the plan being considered and the different possible variable instantiations for it. The domain for all the examples in this paper is naval operations, where the agent is assumed to be a naval officer and the expert a cooperative interface to a fleet information system. The examples assume a scenario in which a particular vessel, the Knox, has been damaged in an accident, thereby lowering its readiness and that of its group. The top-level goal is thus assumed to be restoring the readiness of that group from its current poor rating to good, expressed as (IncreaseGroupReadiness Knox-group poor good).

The domain plans in Pragma are organized in a classification hierarchy based on their effects and preconditions, so that a node in that hierarchy like the top-level instance of IncreaseGroupReadiness in the examples actually stands for the class of plans that would achieve that result in a certain class of situations. The plan class nodes in this hierarchy can thus be used to represent partially specified plans, the set of plans that an agent might be considering that achieves a particular goal using a particular strategy. The subplans (really plan subclasses) of IncreaseGroupReadiness shown in Figure 7 give an idea of the different strategies that the agent may consider for achieving this goal. (Variables are shown with a prefixed question mark.) The plan classification depends on the circumstances, so that RepairShip only functions as a subplan of IncreaseGroupReadiness when its object ship is specified as the Knox, the damaged one, but some of the plans also introduce new variables like ?new-ship, introduced by the ReplaceShip plan, that can take on any

value permitted by the plan's preconditions. Each of these plans also has its own subactions describing how it can be achieved, so that ReplaceShip, for example, involves sailing the ?new-ship to the location of the damaged ship, having it take over the duties of the damaged ship, and then sailing or towing the damaged one to a repair facility.

```
(IncreaseGroupReadiness
    Knox-group poor good)          (1)
 (RepairShip Knox)                 (2)
 (ReinforceGroup Knox-group ?new-ship)  (3)
 (ReplaceShip Knox ?new-ship)      (4)
```

Figure 7: Subplans of IncreaseGroupReadiness

Those subactions, in turn, specify goals for which there can be multiple subplans. The metaplan structures modeling the problem-solving discourse are built on top of this tree of domain plans and actions.

5.2.1 Plan Refining Metaplans

The *build-plan* metaplan is used to capture the agent's goal of constructing a plan to achieve a particular goal, with the *build-subplan* and *build-subaction* metaplans modeling the problem-solving steps that the agent uses to explore and refine the class of domain plans for that goal. An instance of *build-subplan*, say, reflects the agent's choice of one of the possible subplan refinements of the current domain plan as the candidate plan to be further explored. For example, the initial context assuming an IncreaseGroupReadiness plan due to damage to the Knox would be represented in our model by the *build-plan* node on line (1) of Figure 8.

```
(build-plan
   (IncreaseGroupReadiness
      Knox-group poor good))                (1)
 (build-subplan
    (IncreaseGroupReadiness ...)
    (ReplaceShip ...))                      (2)
  (build-plan
     (ReplaceShip Knox ?new-ship))          (3)
   (build-subaction
      (ReplaceShip ...) (Sail ...))         (4)
    (build-plan
       (Sail ?new-ship ?loc Knox-loc))      (5)
```

Figure 8: *Build-Plan*, *Build-Subplan*, and *Build-Subaction*

If we suppose that the agent first considers replacing Knox with some other frigate, that would be modeled as a *build-subplan* child (2) of the *build-plan* for the IncreaseGroup-Readiness plan (1), that would in turn generate a new *build-plan* for ReplaceShip (3). If the agent continues by considering how to get the new ship to that location, that would be represented as a *build-subaction* child (4) of the *build-plan* for ReplaceShip that expands the Sail action.

5.2.2 Variable Constraining Metaplans

In addition to the plan-refining choice of subplans and exploration of subactions, the other plan-building task is the instantiation of the free variables found in the plans. Such variables may either be directly instantiated to a specified value, as modeled by the *instantiate-var* metaplan, or more gradually constrained to subsets of the possible values, as modeled by *add-constraint*.

The *instantiate-var* metaplan reflects the agent's choice of a particular entity to instantiate an open variable in the current plan. For example, the ReplaceShip plan in Figure 8 (3) introduces a free variable for the ?new-ship. If the agent were to choose the Roark as a replacement vessel, that would be modeled by an *instantiate-var* metaplan attached to the *build-plan* node

that first introduced the variable, as shown in Figure 9.

```
(build-plan (ReplaceShip Knox ?new-ship))          (1)
  (instantiate-var ?new-ship Roark)                (2)
    (build-plan (ReplaceShip Knox Roark))          (3)
```

Figure 9: *Instantiate-Var*

The agent may also constrain the possible values for a free variable without instantiating it by using a predicate to filter the set of possible fillers. For example, the agent might decide to consider as replacement vessels only those that are within 500 miles of the damaged one. The predicate from the *add-constraint* node in line (2) of Figure 10 is inherited by the lower *build-plan* node (3), which thus represents the agent's consideration of the smaller class of plans where the value of ?new-ship satisfies the added constraint.

```
(build-plan
   (ReplaceShip Knox ?new-ship))                   (1)
  (add-constraint
     ?new-ship
     (< (distance Knox ?new-ship) 500))            (2)
    (build-plan
       (ReplaceShip Knox ?new-ship))               (3)
```

Figure 10: *Add-Constraint*

The metaplan context tree thus inherits its basic structure from the domain plans as reflected in the *build-plan*, *build-subplan*, and *build-subaction* nodes, and as further specified by the instantiation of domain plan variables recorded in *instantiate-var* and *add-constraint* nodes. Because the domain plans occur as arguments to the plan-building metaplans, the metaplan tree turns out to include all the information that would be available from a normal domain plan context tree, so that no separate domain tree structure is needed.

5.3 Query Metaplans

Although the plan-building metaplans that model the exploration of possible plans and the gradual refinement of an intended plan represent the agent's underlying intent, such moves are seldom observed directly in the expert advising setting. The agent's main observable actions are queries of various sorts, requests for information to guide the plan-building choices. While these queries do not directly add structure to the domain plan being considered, they do provide the expert with indirect evidence as to the plan-building choices the agent is considering. A key advantage of the metaplan approach is the precision with which it models the space of possible queries motivated by a given plan-building context, which in turn makes it easier to predict underlying plan-building structure based on the observed queries. The query metaplans include both plan feasibility queries about plan preconditions and slot data queries that ask about the possible fillers for free variables.

5.3.1 Plan Feasibility Queries

The simplest feasibility query metaplan is *ask-pred-value*, which models at any *build-plan* node a query for a relevant value from one of the preconditions of that domain plan. For example, recalling the original IncreaseGroupReadiness context in which the Knox had been damaged, if the agent's first query in that context is "Where is Knox?", the expert's task becomes to extend the context model in a way that explains the occurrence of that query. While that search would need to explore various paths, one match can be found by applying the sequence of metaplans shown in Figure 11.

```
(build-plan
   (IncreaseGroupReadiness
      Knox-group poor good))                    (1)
  (build-subplan
     (IncreaseGroupReadiness ...)
     (ReplaceShip ...))                         (2)
    (build-plan
       (ReplaceShip Knox ?new-ship))            (3)
      (ask-pred-value
         (ReplaceShip Knox ?new-ship)
         (location-of Knox Knox-loc))           (4)
```

Figure 11: *Ask-Pred-Value*

The *build-subplan* (2) and *build-plan* (3) nodes, as before, model the agent's choice to consider replacing the damaged ship. Because the ReplaceShip domain plan includes among its preconditions (not shown here) a predicate for the location of the damaged ship as the destination for the replacement, the *ask-pred-value* metaplan (4) can then match this query, explaining the agent's question as occasioned by exploration of the ReplaceShip plan. Clearly, there may in general be many metaplan derivations that can justify a given query. In this example, the RepairShip plan might also refer to the location of the damaged ship as the destination for transporting spare parts, so that this query might also arise from consideration of that plan. Use of such a model thus requires heuristic methods for maintaining and ranking alternative paths, but those are not described here.

The other type of plan feasibility query is *check-pred-value*, where the agent asks a yes/no query about the value of a precondition. As an example of that in a context that also happens to require a deeper search than the previous example, suppose the agent followed the previous query with "Is Roark in the Suez?". Figure 12 shows one branch the search would follow, building down from the *build-plan* for ReplaceShip in Figure 11 (3).

```
(build-plan
   (ReplaceShip Knox ?new-ship))                (1)
  (instantiate-var
     (ReplaceShip Knox ?new-ship)
     ?new-ship Roark)                           (2)
    (build-plan
       (ReplaceShip Knox Roark))                (3)
      (build-subaction
         (ReplaceShip ...) (Sail ...))          (4)
        (build-plan
           (Sail Roark Roark-loc Knox-loc))     (5)
          (check-pred-value
             (Sail Roark Roark-loc Knox-loc)
             (location-of Roark Roark-loc))     (6)
```

Figure 12: *Instantiate-Var* and *Build-Subaction*

Here the search has to go through *instantiate-var* and *build-subaction* steps. The ReplaceShip plan has a subaction (Sail ?ship ?old-loc ?new-loc) with a precondition (location-of ?ship ?old-loc) that can match the condition tested in the query. However, if the existing *build-plan* node (1) were directly expanded by *build-subaction* to a *build-plan* for Sail, the ?new-ship variable would not be bound, so that that path would not fully explain the given query. The expert instead must deduce that the agent is considering the Roark as an instantiation for ReplaceShip's ?new-ship, with an *instantiate-var* plan (2) modeling that tentative instantiation and producing a *build-plan* for ReplaceShip (3) where the ?new-ship variable is properly instantiated so that its Sail *sub-action* (5) predicts the actual query correctly.

5.3.2 Slot Data Queries

While the feasibility queries ask about the values of plan preconditions, the slot data queries gather data about the possible values of a free plan variable. The most frequent of the slot data query metaplans is *ask-fillers*, which asks for a list of the items that are of the correct type and that satisfy some subset of the precondition require-

ments that apply to the filler of the free variable. For example, an *ask-fillers* node attached beneath the *build-plan* for ReplaceShip in Figure 12 (1) could model queries like "List the frigates." or "List the C1 frigates.", since the ?new-ship variable is required by the preconditions of ReplaceShip to be a frigate in the top readiness condition.

An *ask-fillers* query can also be applied to a context already restricted by an *add-constraint* metaplan to match a query that imposes a restriction not found in the plan preconditions. Thus the *ask-fillers* node in line (4) of Figure 13 would match the query "List the C1 frigates that are less than 500 miles from the Knox." since it is applied to a *build-plan* node that already inherits that added distance constraint.

```
(build-plan
   (ReplaceShip Knox ?new-ship))                    (1)
 (add-constraint
    ?new-ship
    (< (distance Knox ?new-ship) 500))              (2)
   (build-plan
      (ReplaceShip Knox ?new-ship))                 (3)
    (ask-fillers
       ?new-ship
       (ReplaceShip Knox ?new-ship))
```

Figure 13: *Ask-Fillers*

Note that it is the query that indicates to the expert that the agent has decided to restrict consideration of possible fillers for the ?new-ship slot to those that are closest and thus can most quickly and cheaply replace the Knox, while the restriction in turn serves to make the query more efficient, since it reduces the number of items that must be included, leaving only those most likely to be useful.

There are three other slot data metaplans that are closely related to *ask-fillers* in that they request information about the set of possible fillers but that do not request that the set be listed in full. The *ask-cardinality* metaplan requests only the size of such a set, as in the query "How many frigates are C1?". Such queries can be easier and quicker to answer than the parallel *ask-fillers* query while still supplying enough information to indicate which planning path is worth pursuing. The *check-cardinality* metaplan covers yes/no queries about the set size, and *ask-existence* covers the bare question whether the given set is empty or not, as in the query "Are there any C1 frigates within 500 miles of Knox?".

In addition to the slot data metaplans that directly represent requests for information, modeling slot data queries requires metaplans that modify the information to be returned from such a query in form or amount. There are three such query modifying metaplans, *limit-cardinality*, *sort-set-by-scalar*, and *ask-attribute-value*. The *limit-cardinality* modifier models a restriction by the agent on the number of values to be returned by an *ask-fillers* query, as in the queries "List 3 of the frigates." or "Name a C1 frigate within 500 miles of Knox.". The *sort-set-by-scalar* metaplan covers cases where the agent requests that the results be sorted based on some scalar function, either one known to be relevant from the plan preconditions or one the agent otherwise believes to be so. The function of *ask-attribute-value* is to request the display of additional information along with the values returned, for example, "List the frigates and how far they are from the Knox.".

These modification metaplans can be combined to model more complex queries. For example, *sort-set-by-scalar* and *ask-attribute-value* are combined in the query "List the C1 frigates in order of decreasing speed showing speed and distance from the Knox.". In the metaplan tree, branches with multiple modifying metaplans show their combined effects in the queries they will match. For example, Figure 14 shows the branch that matches the query "What are the 3 fastest frigates?". The

sort-set-by-scalar metaplan in line (2) requests the sorting of the possible fillers of the ?new-ship slot on the basis of descending speed, and the *limit-cardinality* metaplan in that context then restricts the answer to the first 3 values on that sorted list.

```
(build-plan
   (ReplaceShip Knox ?new-ship))                    (1)
 (sort-set-by-scalar
    ?new-ship
    (speed-of ?new-ship ?speed)
    descending)                                     (2)
  (limit-cardinality ?new-ship 3)                   (3)
   (ask-fillers
      ?new-ship
      (ReplaceShip Knox ?new-ship))                 (4)
```

Figure 14: *Sort-Set-by-Scalar and Limit-Cardinality*

As shown in these examples, the slot data query metaplans provide a model for some of the rich space of possible queries that the agent can use to get suggestions of possible fillers. Along with the plan feasibility metaplans, they model the structure of possible queries in their relationship to the agent's plan-refining and variable-instantiating moves. This tight modeling of that connection makes it possible to predict what queries might follow from a particular plan-building path and therefore also to track more accurately, given the queries, which plan-building paths the agent is actually considering.

5.4 Comparison with Other Plan-Based Discourse Models

The use of plans to model the domain task level organization of discourse goes back to Grosz's (1977) use of a hierarchy of focus spaces derived from a task model to understand anaphora. Robinson (1980a, 1980b) subsequently used task model trees of goals and actions to interpret vague verb phrases. Some of the basic heuristics for plan recognition and plan tracking were formalized by Allen and Perrault (1980), who used their plan model of the agent's goals to provide information beyond the direct answer to the agent's query. Carberry (1983, 1984, 1985a, 1985b) has extended that into a plan-tracking model for use in interpreting pragmatic ill-formedness and intersentential ellipsis. The approach presented here builds on those uses of plans for task modeling, but adds a layer modeling problem-solving structure. One result is that the connection between queries and plans that is implemented in those approaches either directly in the system code or in sets of inference rules is implemented here by the query metaplans. Recently, Kautz (1985) has outlined a logical theory for plan tracking that makes use of a classification of plans based on their included actions. His work suggested the structure of plan classes based on effects and preconditions that is used here to represent the agent's partially specified plan during the problem-solving dialogue.

Domain plan models have also been used as elements within more complete discourse models. Carberry's model includes, along with the plan tree, a stack that records the discourse context and that she uses for predicting the discourse goals like *accept-question* or *express-surprise* that are appropriate in a given discourse state. Sidner (1983, 1985) has developed a theory of "plan parsing" for distinguishing which of the plans that the speaker has in mind are plans that the speaker also intends the hearer to recognize in order to produce the intended response. Grosz and Sidner (1985) together have recently outlined a three-part model for discourse context; in their terms, plan models capture part of the intentional structure of the discourse. The metaplan model presented here tries to capture more of that intentional structure than strictly domain plan models, rather than to be a complete model of discourse context.

The addition of metaplans to plan-based models owes much to the work of Wilensky (1983), who proposed a model in which metaplans, with other plans as arguments, were used to capture higher levels of organization in behavior like combining two different plans where some steps overlap. Wilensky's metaplans could be nested arbitrarily deeply, providing both a rich and extensive modeling tool. Litman (1985) applied metaplanning to model discourse structures like interruptions and clarification subdialogues using a stack of metaplan contexts. The approach taken here is similar to Litman's in using a metaplan component to enhance a plan-based discourse model, but the metaplans here are used for a different purpose, to model the particular strategies that shape problem-solving discourse. Instead of a small number of metaplans used to represent changes in focus among domain plans, we have a larger set modeling the problem-solving and query strategies by which the agent builds a domain plan.

Because this model uses its metaplans to capture different aspects of discourse structure than those modeled by Litman's, it also predicts other aspects of agent problem-solving behavior. Because it predicts which queries can be generated by considering particular plans, it can deduce the most closely related domain plan that could motivate a particular query. For instance, when the agent asked about frigates within 500 miles of Knox, the constraint on distance from Knox suggested that the agent was considering the ReplaceShip plan; a similar constraint on distance from port would suggest a RepairShip plan, looking for a ship to transport replacement parts to the damaged one. Another advantage of modeling this level of structure is that the metaplan nodes capture the stack of contexts on which follow-on queries might be based. In this example, follow-on queries might add a new constraint like "with fuel at 80% of capacity" as a child of the existing *add-constraint* node, add an alternative constraint like "within 1000 miles of Knox" as a sibling, query some other predicate within ReplaceShip, or attach even further up the tree. As pointed out below in Section 6, the metaplan structures presented here can also be extended to model alternate problem-solving strategies like *compare-plan* vs. *build-plan*, thus improving their predictive power through sensitivity to different typical patterns of agent movement within the metaplan tree. The clear representation of the problem-solving structure offered in this model also provides the right hooks for attaching heuristic weights to guide the plan tracking system to the most likely plan context match for each new input. Within problem-solving settings, a model that captures this level of discourse structure therefore strengthens an NL system's abilities to track the agent's plans and predict likely queries.

5.5 Applications and Implementation

This improved ability of the metaplan model to track the agent's problem-solving process and predict likely next moves could be applied in many of the same contexts in which domain plan models have been employed, including anaphora and ellipsis processing and generating cooperative responses. For example, consider the following dialogue where the cruiser Biddle has had an equipment failure:

Agent:	Which other cruisers are in the Indian Ocean?	(1)
Expert:	<Lists 6 cruisers>	(2)
Agent:	Any within 200 miles of Biddle?	(3)
Expert:	Home and Belknap.	(4)
Agent:	Any of them at Diego Garcia?	(5)
Expert:	Yes, Dale, and there is a supply flight going out to Biddle tonight.	(6)

The agent first asks about other cruisers that

may have the relevant spare parts. The expert can deduce from the query in line (3) that the agent is considering SupplySparePartByShip. The "them" in the next query in line (5) could refer either to all six cruisers or to just the two listed in (4). Because the model does not predict the Diego Garcia query as relevant to the current plan context, it is recognized after search in the metaplan tree as due instead to a SupplyPartByPlane plan, with the change in plan context implying the correct resolution of the anaphora and also suggesting the addition of the helpful information in (6). The metaplan model of the pragmatic context thus enables the NL processing to be more robust and cooperative.

The Pragma system in which this metaplan model is being developed and tested makes use of the pragmatic model's predictions for suggesting corrections to ill-formed input. Given a suitable library of domain plans and an initial context, Pragma can expand its metaplan tree under heuristic control identifying nodes that match each new query in a coherent problem-solving dialogue and thereby building up a model of the agent's problem-solving behavior. A domain plan library for a subset of naval fleet operations plans and sets of examples in that domain have been built and tested. The resulting model has been used experimentally for dealing with input that is ill-formed due to a single localized error. Such queries can be represented as underspecified logical forms containing "wildcard" terms whose meaning is unknown due to the ill-formedness. By searching the metaplan tree for queries coherently related to the previous context, suggested fillers can be found for the unknown wildcards. For the roughly 20 examples worked with so far, Pragma returns between 1 and 3 suggested corrections for the ill-formed element in each sentence, found by searching for matching queries in its metaplan context model.

5.6 Extensions to the Model and Areas for Further Work

This effort to capture further levels of structure in order to better model and predict the agent's behavior needs to be extended both to achieve further coverage of the expert advising domain and to develop models on the same level for other discourse settings. The current model also includes simplifying assumptions about agent knowledge and cooperativity that should be relaxed.

Within the expert advising domain, further classes of metaplans are required to cover informing and evaluative behavior. While the expert can usually deduce the agent's plan-building progress from the queries, there are cases where that is not true. For example, an agent who was told that the nearest C1 frigate was the Wilson might respond "I don't want to use it.", a problem-solving move whose goal is to help the expert track the agent's planning correctly, predicting queries about other ships rather than further exploration of that branch. Informing metaplans would model such actions whose purpose is to inform the expert about the agent's goals or constraints in order to facilitate the expert's plan tracking. Evaluative metaplans would capture queries whose purpose was not just establishing plan feasibility but comparing the cost of different feasible plans. Such queries can involve factors like fuel consumption rates that are not strictly plan preconditions. The typical patterns of movement in the metaplan tree are also different for evaluation, where the agent may compare two differently-instantiated *build-plan* nodes point for point, moving back and forth repeatedly, rather than following the typical feasibility pattern of depth-first exploration. Such a comparison pattern is highly structured, even though it would appear to the current model as pat-

temless alternation between *ask-pred-value* queries on two different plan branches. Metaplans that capture that layer of problem-solving strategy would thus significantly extend the power of the model.

Another important extension would be to work out the metaplan structure of other discourse settings. For an example closely related to expert advising, consider two people trying to work out a plan for a common goal; each one makes points in their discussion based on features of the possible plan classes, and the relationship between their statements and the plans and the strategy of their movements in the plan tree could be formalized in a similar system of metaplans.

The current model also depends on a number of simplifying assumptions about the cooperativeness and knowledge of the agent and expert that should be relaxed to increase its generality. For example, the model assumes that both the expert and the agent have complete and accurate knowledge of the plans and their preconditions. As Pollack (1986) has shown, the agent's plan knowledge should instead be formulated in terms of the individual beliefs that define what it means to have a plan, so the model can handle cases where the agent's plans are incomplete or incorrect. Such a model of the agent's beliefs could also be a major factor in the heuristics of plan tracking, identifying, for example, predicates whose value the agent does not already know which therefore are more likely to be queried. The current model should also be extended to handle multiple goals on the agent's part, examples where the expert does not know in advance the agent's top-level goal, and cases of interactions between plans.

However, no matter how powerful the pragmatic modeling approach becomes, there is a practical limitation in the problem-solving setting on the amount of data available to the expert in the agent's queries. More powerful, higher level models require that the expert have appropriately more data about the agent's goals and problem-solving state. That tradeoff explains why an advisor who is also a friend can often be much more helpful than an anonymous expert whose domain knowledge may be similar but whose knowledge of the agent's goals and state is weaker. The goal for cooperative interfaces must be a flexible level of pragmatic modeling that can take full advantage of all the available knowledge about the agent and the recognizable elements of discourse structure while still avoiding having to create high-level structures for which the data is not available.

References

[1] James F. Allen and C. Raymond Perrault. Analyzing Intention in Utterances. *Artificial Intelligence* 15:143-178, 1980.

[2] Sandra Carberry. Tracking User Goals in an Information-Seeking Environment. In *Proceedings of the National Conference on Artificial Intelligence*, pages 59-63. William Kaufmann, 1983.

[3] Sandra Carberry. Understanding Pragmatically Ill-Formed Input. In *Proceedings of the 10th International Conference on Computational Linguistics*, pages 200-206. Association for Computational Linguistics, 1984.

[4] Sandra Carberry. A Pragmatics-Based Approach to Understanding Intersentential Ellipsis. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 188-197. 1985.

[5] M. Sandra Carberry. *Pragmatic Modeling in Information System Interfaces.* PhD thesis, University of Delaware, 1985.

[6] Grosz, B.J. and Sidner, C.L. *The Structures of Discourse Structure.* Technical Report 6097, Bolt Beranek and Newman, 1985.

[7] Barbara J. Grosz. The Representation and Use of Focus in a System for Understanding Dialogues. In *Proceedings of the International Joint Conference on Artificial Intelligence 77*, pages 67-76. Morgan Kaufmann, 1977.

[8] Henry A. Kautz. *Toward a Theory of Plan Recognition.* Technical Report TR 162, University of Rochester, July, 1985.

[9] Diane J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues.* PhD thesis, University of Rochester, 1985.

[10] Martha E. Pollack. *Inferring Domain Plans in Question-Answering.* PhD thesis, University of Pennsylvania, 1986.

[11] Ann E. Robinson. *The Interpretation of Verb Phrases in Dialog.* Technical Report 206, SRI International, 1980.

[12] Ann E. Robinson, Douglas E. Appelt, Barbara J. Grosz, Gary G. Hendrix, and Jane J. Robinson. *Interpreting Natural-Language Utterances in Dialogs About Tasks.* Technical Report 210, SRI International, 1980.

[13] Candace L. Sidner. What the Speaker Means: The Recognition of Speaker's Plans in Discourse. *International Journal of Computers and Mathematics* 9(1):71-82, 1983.

[14] Candace L. Sidner. Plan Parsing for Intended Response Recognition in Discourse. *Computational Intelligence* 1:1-10, 1985.

[15] Robert Wilensky. *Planning and Understanding.* Addison-Wesley, 1983.

6. Rapid Porting of the Parlance™ Natural Language Interface

Madeleine Bates[9]

Abstract

Developing knowledge bases for AI systems takes too long and costs too much. Even a "portable" system is expensive to use if its installation takes a long time or requires the labor of scarce, highly-trained people. BBN has recently created a tool for acquisition which dramatically reduces the time and cost of installing a natural language system.

During 1988, BBN used its Learner™ tool to configure the Parlance™ database interface to two different versions of a large Navy database. The configuration process was performed primarily with development versions of the Learner, which is a software tool for creating the knowledge bases, vocabulary, and mappings to the database that enable the Parlance interface to understand questions addressed to a particular database. The Learner reduced the time required to create Parlance configurations fr om months to weeks, and demonstrated that the Learner works effectively on databases with many hundreds of fields.

6.1 Introduction

6.1.1 The Navy's IDB Database

The IDB (hereafter called the Navy database) is a large, evolving database being used in the Fleet Command Center at the Navy's Pacific Fleet headquarters in Pearl Harbor, Hawaii [1]. It has dozens of tables and hundreds of fields containing information about hundreds of U.S. ships, planes and other units, as well as more limited data on foreign units.

Examples of the kind of information that may be available for a particular unit are: its home port, current location, current employments (an employment is a complex concept including destination, projected arrival time, purpose, etc.), type and amount of equipment on board, various types of readiness status (personnel readiness, equipment readiness, overall readiness, etc.), and operating characteristics (average cruising speed, maximum speed, fuel capacity, etc.). Other data in this database include deta iled information about the characteristics of various types of equipment (e.g., the firing rate of guns) and properties of geographic entities (e.g., for ports, the country they are in, and whether they have a deep channel).

The Navy database provides basic data for systems under development at the Fleet Command Center. This database offers a rich environment for a natural language in-

[9]This paper is a reprint of a paper that appears in **Proceedings of the Speech and Natural Language Workshop**, Morgan Kaufmann Publishers, Inc., San Mateo, CA, February, 1989.

terface, because the need to explore the database with ad hoc queries occurs frequently.

6.1.2 The Parlance Interface

The Parlance interface from BBN Systems and Technologiés Corporation is an English language database front end. It has a number of component parts: a graphical user interface, a language understander that translates English queries into database commands for relational database systems such as Oracle and VAX Rdb, a control structure for interacting with the user to clarify ambiguous queries or unknown words, and a dbms driver to call the database system to execute database commands and to return retriev ed data to the user.

The Parlance system uses several domain-dependent knowledge bases:

1. A domain model, which is a class-and-attribute representation of the concepts and relationships that the Parlance user might employ in queries.
2. A mapping from this domain model to the database, which specifies how to find particular classes and attributes in terms of the database tables and fields of the underlying dbms.
3. A vocabulary, containing the lexical syntax and semantics of words and phrases that someone might use to talk about the classes and attributes.
4. Miscellaneous additional information about how information is to be printed out (for example, column headers that are different from field names in the database).

The Learner is used to create these knowledge bases.

The following queries illustrate the kinds of questions that one can ask the Parlance system after it is configured for the Navy database:

What's the maximum beam of the Kitty Hawk?

Show me the ships with a personnel resource readiness of C3.1

List the ships that are C1 or C2.

Is the Frederick conducting ISE in San Diego?

How many ships aren't NTDS capable?

Which classes have a larger fuel capacity than the Wichita?

How many submarines are in each geo region.

Are there any harpoon capable C1 ships deployed in the Indian Ocean whose ASW rating is M1?

List them.

Show the current employment of the carriers that are C3 or worse, sorted by overall readiness.

Where is the Carl Vinson?[10]

What are the positions of the friendly subs?

6.1.3 The Learner

The Learner is a software tool that creates the domain-dependent knowledge bases that the Parlance system needs. It "learns" what Parlance needs to know from several sources:

[10]This query is ambiguous. It may be asking for a geographical region or for a latitude and longitude. The Parlance system recognizes the ambiguity and asks the user for clarification.

1. The database system itself (i.e., the dbms catalogue that describes the database structure, and the values in various fields of the database).
2. A human teacher (who is probably a database administrator, someone familiar with the structure of the database, but who is not a computational linguist or AI expert).
3. A core domain model and vocabulary that are part of the basic Parlance system.
4. Inferences (about such things as morphological and syntactic features) that the Learner makes (subject to correction and modification by the teacher).

Figure 1 shows the input and output structure of the Learner. We call the process of using the Learner *configuring* Parlance for a particular application.

The human teacher uses the Learner by stepping through a series of menus and structured forms. The Learner incrementally builds a structure that can be output as the knowledge bases shown in Figure 1.

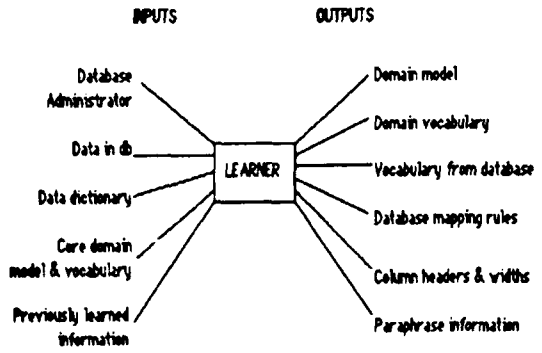

Figure 1. The Structure of the Learner

The teacher chooses particular actions and is led through steps which elicit related information that Parlance must know. For example, when the teacher designates that a particular table or set of tables belong to a class named "ship", the Learner immediately allows the teacher to give synonyms for this class, such as "vessel". The Learner will then infer that the plural form of the synonym is "vessels", instead of making the teacher supply the plural form, although the teacher can easily correct the Learner if the word has an irregular plural.

Whenever information is optional, the teacher can decline to specify it at the first opportunity, and can later initiate an action to provide it. Both required and optional information can be changed by the teacher using the Learner's editing capabilities.

The ability to assign names freely, the freedom to do many operations in the sequence that makes the most sense to the person using the Learner, and the fact that the Learner expresses instructions and choices in database terms wherever possible, make it easy for database administrators who are not computational linguists or AI experts to configure the Parlance interface.

6.2 Configuring Parlance

Before the Learner existed, Parlance configurations were created "by hand". That is, highly skilled personnel had to use a separate set of programs (including a Lisp editor) to create the appropriate configuration files.

Figure 2 compares this by-hand configuration process with the first experience using the Learner on the Navy database. The two examples used different databases, but in each case we began with a large set of sample queries in the target domain, and periodically tested the developing configuration by running those queries through the Parlance system. We measured our progress by keeping track of the number of those

queries the system could understand as the configuration process went on. Figure 2 actually considerably understates the productivity enhancement realized with the Learner, because the personnel database used for the by-hand configuration was much smaller and less complex than the Navy database.

The Navy database used to test the first version of the Learner was considerably restructured and enlarged, and we had an opportunity to configure Parlance for the newer database. Since we had a new, improved version of the Learner, we chose to configure Parlance to the second version of the Navy database "from scratch", rather than by building on the results of the first configuration. This gave us an opportunity to measure the effort required to use the Learner to do a much larger system configuration , since the size of the target database (measured in terms of the number of fields) had nearly tripled.

The results in Figure 3 and its accompanying notes show that the Learner robustly scaled up to the task, and that the time required to perform the configuration increased much less than the number of fields in the database, the vocabulary size, or any other simple metric of size. In fact, for a modest 1/3 increase in configuring effort, a configuration roughly 3 times larger was created.

Notes to accompany Figure 3:

(0) Changes in the underlying system since this configuration was created make it impossible to measure some of the numbers in this column accurately, so the numbers dealing with vocabulary are estimates.

(1) Records were not kept at the time this configuration was created, but the configuration happened over a period of months.

(2) That this level of effort includes not just time spent using the Learner but also time required to understand the domain, and to do some testing and revision. About 60% of this time was spent using the development version of the Learner.

(3) Records were not kept at the time this configuration was done, but it involved many person-months.

(4) This estimate includes inflected forms of regular words and some words that were acquired directly from database fields.

(5) This includes words read from the database and all words directly represented in the vocabulary; it excludes inflected forms of morphologically regular words.

(6) This is a rough measure of the semantic complexity of the domain, since it excludes words that are abbreviations or synonyms.

6.3 Conclusions

The Learner[11] significantly reduces the time required to create configurations of the Parlance natural language interface for databases with hundreds of fields from months to weeks. This dramatic speed-up in knowledge acquisition scales up robustly, and works as effectively on large databases as it does on small ones.

References

[11]Parlance and Learner are trademarks of Bolt Beranek and Newman Inc. or its subsidiaries. Vax and Rdb are trademarks of Digital Equipment Corp. Oracle is a trademark of Oracle Corp.

[1] Ceruti, M.G., Schill, J.P. *FCCBMP Data Dictionary, Version 3.* Technical Report, Naval Ocean Systems Center, Code 423, San Diego, CA, 1988.

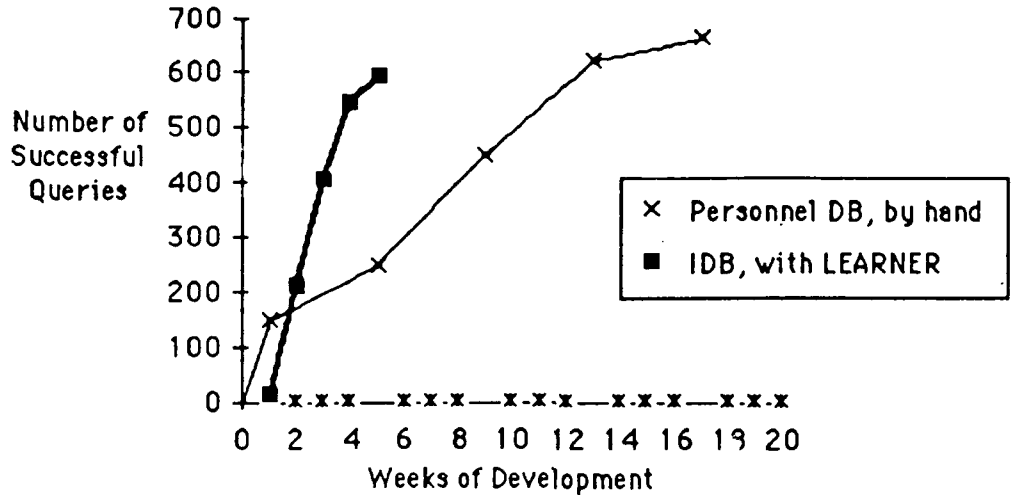

Figure 2. Speed-up of acquisition using the Learner

	Personnel Configuration (0)	1st Navy Configuration	2nd Navy Configuration
Elapsed time	(1)	4 weeks	6 weeks
Total level of effort (2)	(3)	6+ per.wks.	8+ per. wks.
Tables in database	31	32	75
Fields in database	133	231	666
Classes in domain model	218	83	303
Attributes in domain model	316	160	680
Estimated total vocabulary (4)	3000	5500	9800
Root forms (5)	1700	3282	6354
Proper nouns read from db	1170	2656	3907
Verbs	65	6	36
Words with semantics (6)	1600	3326	6073

Figure 3. Comparing the Configuration Processes and Results

7. Strategies for Effective Paraphrasing

Marie Meteer, Varda Shaked[12]

Abstract

In this paper we present a new dimension to paraphrasing text in which characteristics of the original text motivate strategies for effective paraphrasing. Our system combines two existing robust components: the IRUS-II NL understanding system and the Spokesman generation system. We describe the architecture of the system and enhancements made to these components to facilitate paraphrasing. We particularly look at how levels of representation in these two systems are used by specialists in the paraphraser which define potential problems and paraphrasing strategies. Finally, we look at the role of paraphrasing in a cooperative dialog system. We will focus here on paraphrasing in the context of natural language interfaces and particularly on how multiple interpretations introduced by various kinds of ambiguity can be constrasted in paraphrases using both sentence structure and highlighting and formating the text itself.

7.1 Introduction[13]

While technically paraphrasing is simply the task of restating the meaning of a text in a different form, it is crucial to consider the purpose of the paraphrase in order to motivate particular strategies for changing the text. If the point of the paraphrase is to clarify the original text, as in a natural language (NL) interface to a database or expert system application, then disambiguating the query and choosing more precise lexical items (perhaps closer to the structure of the actual DB, expert system, or other underlying applicatiton) are essential strategies. If the point is to summarize information, then strategies for evaluating the relative importance of the information presented in the text is necessary. If the point is merely to restate the text differently than the original, perhaps merely to excercise the system, then one must use strategies which consider what structures and lexical items were actually found by the parser.

[13]We would like to thank Lance Ramshaw for his invaluable help in understanding the inner workings of RUS and suggestions of where it could be augmented for our purposes, and Dawn MacLaughlin for her implementation of Parrot, the initial version of our paraphraser. We would also like to thank Ralph Weischedel, Damaris Ayuso, and David McDonald for their helpful comments of drafts of this paper and Lyn Bates for early inspirations.

[12]This paper is reprinted from the *Proceedings of the 27th Annual Meeting of the 12th International Conference on Compuational Linguistics*, 22-27 August 1988, Budapest Hungary. Requests for copies should be addressed to:

Dr. Donald E. Walker (ACL)
Bell Communications Research
435 South Street MRE 2A379
Morristown, NJ 07960, USA

Our motivation for work on strategies for effective paraphrasing comes from the recent availablility of NL interfaces as commercial products. As the underlying systems that a NL interface must interact with increase in number and sophistication, the range of NL interactions will increase as well. Paraphrasers developed in the past (e.g. McKeown's Co-op and Bates & Bobrow's Parlance™ software) were all limited in that each used only a single strategy for paraphrasing regardless of what problems may have been present in the original query. (We discuss these systems in detail in Section 6.) Our approach is to develop a variety of strategies which may be employed in different situations. We introduce a new dimension to paraphrasing text in which characteristics of the original text plus the overall context (including the goal of the system) motivate strategies for effective paraphrasing.

Our focus here will be on paraphrasing ambiguous queries in an interactive dialog system, where contrasting multiple interpretations is essential. In order to ground our discussion, we first look briefly at a range of ambiguity types. We then provide an overview of the architecture and description of the two major components: the IRUS-II™ understanding system and the Spokesman generation system. We look closely at the aspects of these systems that we augmented for the paraphrasing task and provide a detailed example of how the system appreciates multiple interpretations and uses that information to govern decision making in generation. Next we discuss the role of paraphrasing in a cooperative dialog system, and in the final section we contrast our approach with other work in paraphrasing.

7.2 Problems and Strategies

Ambiguity is one of the more difficult problems to detect and correct. In this section we look at three kinds of ambiguity: lexical, structural and contextual, and discuss potential strategies a paraphraser might use to eliminate the ambiguity.

1) LEXICAL AMBIGUITIES are introduced when a lexical item can refer to more than one thing. In the following example "Manhattan" can refer to either the borough of New York City or the ship:

What is the latitude and longitude of Manhattan'

The paraphraser must appreciate the ambiguity of that noun phrase, decide how to disambiguate it, and decide how much of the context to include in the paraphrase. One strategy would be to repeat the entire query, disambiguating the noun phrase by using the type and name of the object.

Do you mean what is the latitude and longitude of the city Manhattan or what is the latitude and longitude of the ship Manhattan'

However, if the query is long, the result could be quite cumbersome. A different strategy, highlighting and formatting the text, can serve to direct the user's attention to the part that is ambiguous:

Do you mean list the latitude and longitude of the city Manhattan or the ship Manhattan'

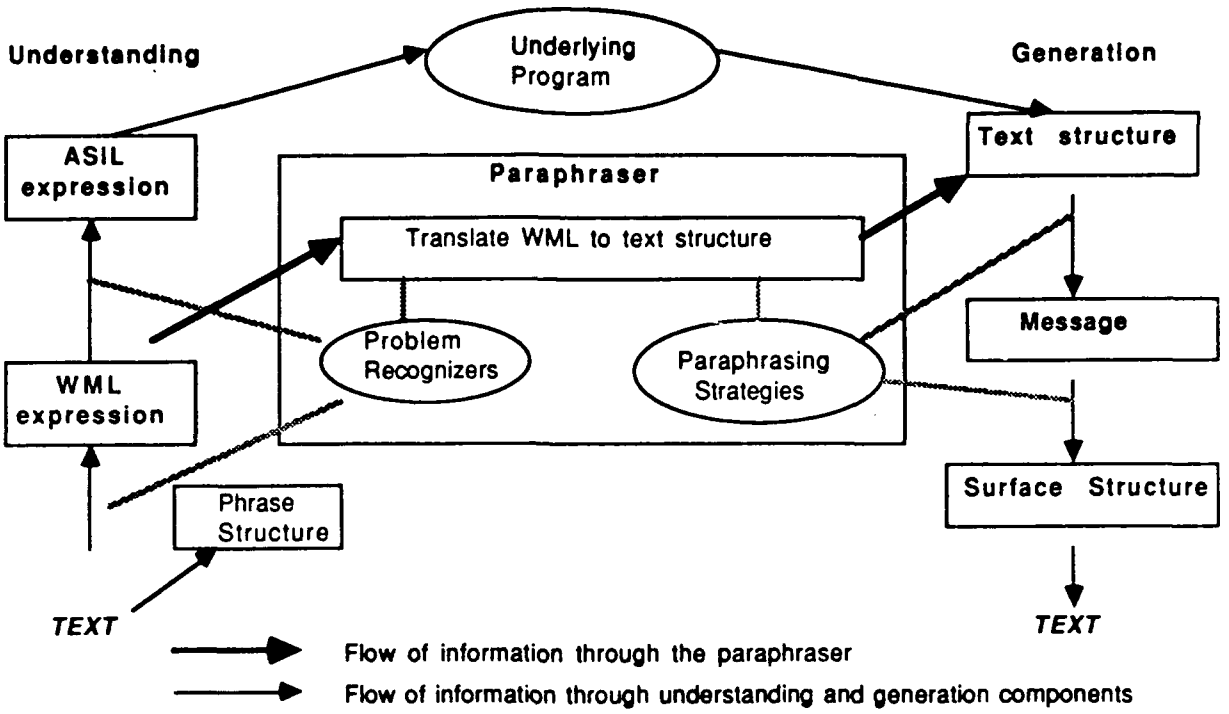

Figure 15: Architecutre of the Paraphraser

2) STRUCTURAL AMBIGUITIES are caused when there are multiple parses for a sentence. Conjunction is a typical source of structural ambiguity. Modifiers of conjoined NPs may distribute over each NP or modify only the closest NP. Consider, for example, the following query:

Display the forested hills and rivers.

This query has only one interpretation in which the premodifier "forested" modifies only the noun "hills". In contrast the following query has two interpretations:

Display the C1 carriers and frigates

In one interpretation, the premodifier "C1" may apply only to the noun "carrier"; in the other, "C1" applies to both "carriers" and "frigates". Each interpretation requires a different paraphrase strategy. In the case where the premodifier distributes, the ambiguity may be eliminated by repeating the modifier: *Display the C1 carriers and C1 frigates*. When it does not distribute, there are three potential strategies:

--changing the order of the conjuncts: *Display the frigates and C1 carriers.*

--introducing explicit quantifiers: *Display the C1 carriers and all the frigates.*

--moving premodifiers to postmodifiers: *Display the carriers which are C1 and the frigates.*

3) CONTEXTUAL AMBIGUITIES are introduced when the query is underspecified for the underlying system it is working with. For example if the context includes a map and the possibility of natural language or table output, the query *Which carriers are C1?* could mean either *list or display*.

7.3 Architecture

As the examples above illustrate, the information needed to notice problems such as ambiguity in a query is quite varied, and the strategies needed to generate a motivated paraphrase must be employed at various levels in the generation process. A distinguishing feature of our system is that it works in cooperation with existing understanding and generation components and allows the paraphraser access to multiple levels of their processing. This multilevel design allows the understanding system to appreciate ambiguities and vaugeness at lexical, structural, and contextual levels, and the generation system to affect the text's organization, syntactic structure, lexical items and even to format and highlight the final text.

Figure 9 shows an overview of the architecture of the system. In this section, we first describe the understanding and generation systems independently, particularly at how the problem recognizers and paraphrasing strategies have been incorporated into the components. We then look at the paraphraser itself and how it evolved.

7.3.1 The Understanding Component: IRUS-II(TM) NL Software

IRUS-II™ (Weischedel, et al. 1987) is a robust natural language understanding system that interfaces to a variety of underlying systems, such as DB management systems, expert systems and other application programs. It is capable of handling a very wide range of English constructions including ill-formed ones.

7.3.1.1 IRUS-II - Components and design principals

IRUS-II has two major processing levels which distinguish the linguistic processing from the details of the particular underlying systems it is used with. The first level, the "Front End", integrates syntactic and semantic processing. The major domain-independent "Front End" modules include a parser and associated grammar of English, a semantic interpreter, and a subsystem for resolving anaphora and ellipsis. These modules simultaneously parse an English text into a syntactic structural description and construct a formal semantic representation of its meaning in a higher order intensional logic language called the World Model Language (WML). The syntactic processor is the RUS Grammar/Parser which is based on the ATN formalism. Constants in the WML are concepts and predicates from a hierarchical domain model represented in NIKL, (Moser 1983).

The more domain dependent modules of the Front End are the lexicon, domain model, and a set of semantic Interpretation Rules (IRules). The lexicon contains the information about parts of speech, and syntactic and morphological features needed for parsing, and word and phrase substitutes (such as abbreviations). An IRule defines, for a word or (semantic) class of words, the semantically acceptable English phrases that can occur having that word as a head of the phrase, and in addition defines the semantic interpretation of an accepted phrase. Thus, when the parser proposes (i.e., TRANSMITs) an intermediate syntactic phrase structure, the semantic interpreter uses the IRules that are associated with the head of that phrase to determine whether the proposed structure is interpretable and to specify its interpretation. Since semantic processing is integrated wth syntactic processing, the IRules serve to block a semantically anomalous phrase as soon as it is proposed by the parser. The semantic representation of a phrase is constructed only when the phrase is believed complete.

The task of the "Back End" component of IRUS is to take a WML expression and compute the correct command or set of commands to one or more underlying systems, obtaining the result requested by the user. This problem is decomposed into the following steps:

- The WML expressions is simplified and then gradually translated into the Application System Interface Langauge (ASIL).
- The particular underlying system or systems that need to be accessed are identified.
- The ASIL is transformed into underlying system(s) code to execute the query.

While the constants in WML and ASIL are domain-dependent, the constants in ASIL-to-code translation system(s) code are both domain dependent and underlying-system dependent.

7.3.1.2 Ambiguity Handling by the IRUS-II system - Overview

In this section, we briefly describe how various kinds of ambiguities are currently handled in IRUS-II. There are at least the following kinds of ambiguities that may occur in natural language: Semantic ambiguity (lexical, phrasal, referring expressions), structural ambiguity, quantifier scope ambiguity and collective reading ambiguity. In cases of semantic ambiguity, multiple WMLs are generated from the same syntactic parse path. For example, when a word (e.g., "Manhattan") belongs to more than one semantic class in the domain model (e.g, CITY, VESSEL), two WMLs are generated from the same syntactic parse

path, each referring to the different semantic class. Similarly, nouns premodified by nouns/adjectives (e.g., "Hawaii ships") generate multiple WMLs, each created as a result of multiple IRules assigning several interpretations to the modified noun (e.g., "Ships whose home port is Hawaii", "Ships whose destination is Hawaii", or "Ships whose current location is Hawaii").

Structural ambiguities are caused by multiple syntactic interpretations and result in alternative parse paths in the RUS parser/grammar. IRUS-II identifies these ambiguities by sequentially attempting to parse the text, with each attempt following a different parse path. Note in these cases each syntactic parse path may also have multiple semantic interpretations.

7.3.1.3 Enhancements to IRUS-II for effective paraphrasing

Though IRUS-II produces multiple interpretations (WMLs) for a variety of ambiguous sentences, it was not originally designed with the intent of paraphrasing those interpretations. While each individual WML could be paraphrased separately, a more useful approach would be to to combine closely related interpretations into a single paraphrase that highlights the contrasts between the interpretations. The need to keep associations between multiple interpretations motivated the following enhancements to the IRUS-II system:

- Predefined ambiguity specialists that detect and annotate potential problems presented by the input text are "distributed" in the parser/grammar and the semantic interpreter. For example, when the parser TRANSMITs the phrase "Manhattan" to the semantic interpreter as a head of a noun phrase (NP) , two semantic classes, CITY and VESSEL, will be associated with that NP. At this point, the Lexical Ambiguity Specialist will record the lexical item "Manhattan" as the ambiguity source and the two different classes.
- After recording the potential ambiguity source, each ambiguity specialist monitors a predefined sequence of TRANSMITs associated with that source, and records the different intermediate WML expressions resulting from these TRANSMITs. For example, the Lexical Ambiguity Specialist monitors the TRANSMITs of "Manhatten" as a head noun of the NP. In this case, there will be two applicable IRules, one defining "Manhattan" as a CITY and the other defining "Manhattan" as a VESSEL. Both interpretations are semantically acceptable, resulting in two intermediate WMLs, which are then recorded by the specialist. Upon completion of the input text, two WMLs will be created and this record will be used to annotate them with their respective differences that resulted from a common ambiguity source.

We look at the details of the specialists on one particular example in Section 4.

7.3.2 The Generation System: SPOKESMAN

The Spokesman generation system also has two major components: a text planner and a linguistic realization component, Mumble-86 (Meteer et al. 1987). Both components are built within the framework of "multilevel, description directed control" (McDonald 1983, McDonald & Pustejovsky 1985). In this framework, decisions are organized into levels according to the kind of reference knowledge brought to bear (e.g. event or argument structure, syntactic structure, morphology). At each level, a representation of the utterance is constructed

which both captures the decisions made so far and constrains the future decision making. The representation at each level also serves as the control for the mapping to the next level of representation.

The text planner must establish what information the utterance is to include and what wording and organization it must have in order to insure that the information is understood with the intended perspectives. The intermediate level of representation in this component is the text structure , which is a tree like representation of the organization of discourse level constituents. The structure is populated with model level objects (e.g. from the applications program) and "discourse objects" (compositional objects created for the particular utterance) and the relations between these objects. The text structure is extended incrementally in two ways:

1) expanding nodes whose contents are composite objects by using predefined templates associated with the object types (such as exanding an "event" object by making its arguments subnodes);

2) adding units into the structure at new nodes. The units may be selected from an already positioned composite unit or they may be individuals handed to the orchestrator by an independently driven selection process.

Once the text structure is complete, it is traversed depth first beginning with the root node. At each node, the mapping process chooses the linguistic resource (lexical item, syntactic relation such as restrictive modifier, etc.) that is to realize the object which is the contents of that node. Templates associated with these objects define the set of possibilities and provide procedures for building its portion of the next level of representation, the "message level", which is the input specification for the linguistic realization component, MUMBLE-86.

The input specification to MUMBLE-86 specifies what is to be said and constrains how it is to be said. MUMBLE-86 handles the realization of the elements in the input specification (e.g. choosing between *the ships are assigned, which are assigned,* or *assigned* depending on whether the linguistic context requires a full clause, postmodifier, or premodifier), the positioning of elements in the text (e.g. choosing where to place an adverbial phrase), and the necessary morphological operations (e.g. subject verb agreement).

In order to make these decisions, MUMBLE-86 maintains an explicit representation of the linguistic context in the form of an annotated surface structure. Labels on positions provide both syntactic constraints for choosing the appropriate phrase and a definition of which links may be broken to add more structure. This structure is traversed depth first as it is built, guiding the further realization of embedded elements and the attachment of new elements. When a word is reached by the traversal process, it is sent to the morphology process, which uses the linguistic context to execute the appropriate morphological operations. Then the word is passed to the word stream to be output and the traversal process continues through the surface structure.

7.3.2.1 Parrot and Polly

Our first implementation of the paraphraser was simply a parrot which used the output of the parser (the WML) as input to the generator. The text planner in this case consists of a set of translation functions which build text structure and populate it with composite objects built from WML subexpressions and the constants in the WML (concepts and roles from IRUS's hierarchical domain model). The translation to text structure uses both explicit and implicit information from the WML. The first

operator in a WML represents the speech act of the utterance. For example, bring-about indicates explicitly that the matrix clause should be a command and implicitly that it should be in the present tense and the agent is the system. The iota operator indicates that the reference is definite and power indicates it is plural.

A second set of templates map these objects to the input specification for the linguistic component, determining the choice of lexical heads, argument structures, and attachment relations (such as restrictive-modifier or clausal-adjunct).

Interestingly, parrot turned out to be a conceptual parrot, rather than a verbatim one. For example, the phrase *the bridge on the river* is interpreted as the following wml expression. The domain model predicate CROSS representing the role between *bridge* and *river* because IRUS interprets "on" in this particular context in terms of the CROSS relation:

(IOTA JX124 BRIDGE (CROSS JX124
(IOTA JX236 RIVER)))

This is "parroted" as *the bridge which crosses the river*. While in some cases this direct translation of the wml produces an acceptable phrase, in other cases the results are less desirable. For example, named objects are represented by an expression of the form (IOTA *var type* (NAME *var name*)), which, translated directly, would produce *the river which is named Hudson*. Such phrases make the generated text unnecessarily cumbersome. Our solution in PARROT was to implement an optimization at the point when the complex object is built and placed in the text structure that uses the name as the head of the complex object rather than the type. (Melish, 1987, discusses similar optimizations in generating from plans.)

While PARROT allowed us to establish a link from text in to text out, it is clear this approach is insufficient to do more sophisticated paraphrasing. POLLY, as we call our "smart" paraphraser, takes advantage of the extra information provided by IRUS-II in order to control the decision making in generation.

One of the most common places in which the system must choose carefully which realization to use is when the input is ambiguous and the paraphrase must contrast the two meanings. For example, if a semantic ambiguity is caused by an ambiguous name, as in *Where is Diego Garcia* (where Diego Garcia is both a submarine and a port), the type information must be included in the paraphrase:

Do you mean where is the port Diego Garcia or the submarine Diego Garcia.

Note, with the optimization of PARROT described above, this sentence could not be disamiguated.

In order to generate this paraphrase contrasting the two interpetations, the system needs to know what part is ambiguous at two different points in the generation process: in the text planner when selecting the information to include (both the type and the name) and at the final stage when the text is being output (to change the font). Our use of explicit active representations allows the system to mark the contrast only once, at the highest level, the text structure. This constraint is then passed through the levels and can effect decisions at any of the lower levels. Thus the system makes use of the information provided by the understanding system when it is available and ensures it will still be available when needed and won't be considered in parts of the utterance where it is not relevant.

7.4 Paraphrasing Syntactic Ambiguities - an Example

To elucidate the description above, we will return to an earlier example of a query with an ambiguous conjunction construction: *Display all carriers and frigates in the Indian Ocean.* This sentence has two possible interpretations:

1) *Display all carriers in the Indian Ocean and all frigates in the Indian Ocean.*

2) *Display all frigates in the Indian Ocean and all the carriers.*

In this example we show (1) how the Problem Recognizers discover that there are two interpretations and what the particular differences are; and (2) how the Paraphrasing Strategies use that information in the translation to text structure and the generation of the paraphrase.

7.4.1 Phase 1: The Problem Recognizers

As we discussed earlier, problem recognizing specialists have been embedded in the understanding system. Here we look at the Noun-Phrase (NP) Conjunction Ambiguity specialist and the two parse paths that correspond to the parses resulting from a NP conjunction ambiguity (see Figure 10).

The first task of this specialist is to annotate the parse path when a NP conjunction is encountered by the parser. In IRUS-II, when the RUS parser has completed the processing of the first NP *the frigates* and the conjunction word *and*, it attempts (among other alternatives) to parse the next phrase as a NP. At this point the Conjunction Ambiguity Specialist annotates that parse path with a NP-CONJUNCTION-AMBIGUITY tag (depicted in Figure 10 with * at the first NPLIST/ state in both parse paths 1 and 2). This annotation will allow the different interpretations that may result from this NP conjunction to be grouped later according to their common ambiguity source. (Note that if not using an ATN, appropriate annotations can be made using structure building rules associated with the grammar rules). The paraphraser can then orgnaize its paraphrases according to a group of related ambiguous interpretations. As previously stated, it is believed that simultaneously observing closely related interpretations is more effective than presenting randomly generated paraphrases that correspond to arbitrary parse paths.

The second task of the NP Conjunction Ambiguity specialist is to monitor those TRANSMITs to the Semantic Interpreter that may result in multiple interpretations (WMLs) from the same source of ambiguity. Thus, starting from when the possible ambiguity has been noticed, this specialist will monitor the TRANSMITs to all the modifiers of the NPs. In our example, the NP Conjunction Ambiguity specialist monitors the TRANSMITs of the prepositional phrase (PP) *in the Indian Ocean* to all

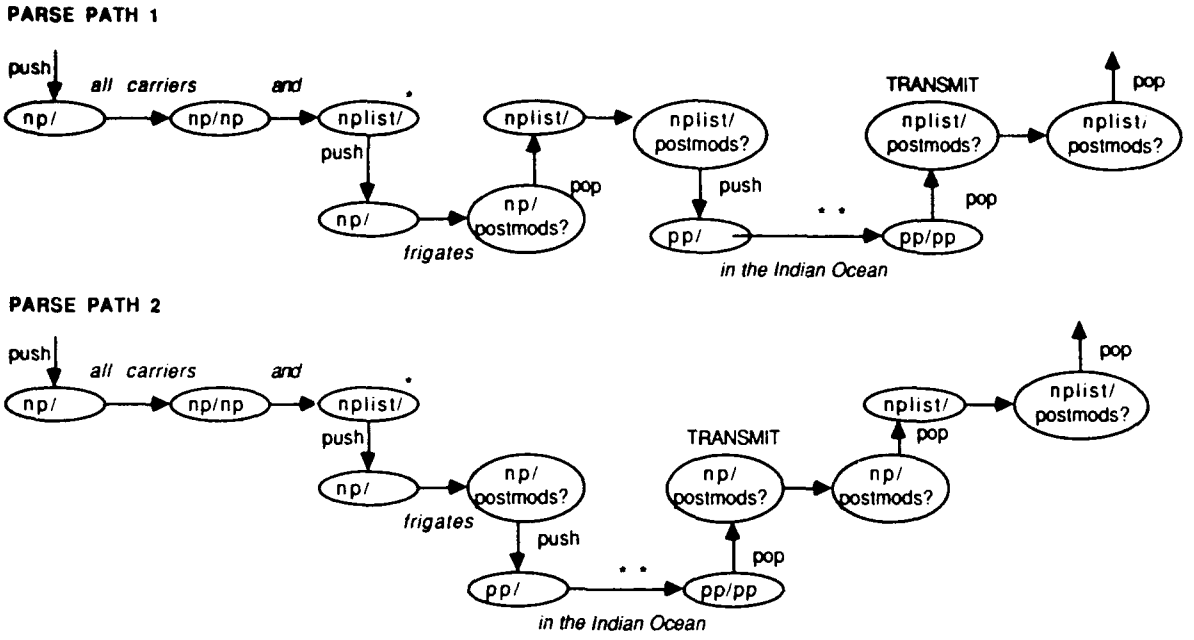

Figure 16: Parse Paths

NPs annotated with the NP-CONJUNCTION-AMBIGUITY tag (TRANSMITs are illustrated with **), which include the TRANSMITs of that PP as a postmodifer to each of the conjoined NPs (parse path 1) as well as to only the second NP (parse path 2). Since the PP *in the Indian Ocean* is semantically acceptable as a postmodifer in both parse paths, two intermediate WMLs will be created:

```
Intermediate WML-1:
 (SETOF (IOTA ?JX19 (POWER CARRIER)
         (UNITS.LOCATION ?JX19 IO))
        (IOTA ?JX20 (POWER FRIGATE)
         (UNITS.LOCATION ?JX20 IO)))

Intermediate WML-2:
(SETOF (IOTA ?JX19 (POWER CARRIER))
       (IOTA ?JX20 (POWER FRIGATE)
        (UNITS.LOCATION ?JX20 IO)))
```

Each intermediate WML contains a SETOF operator with two arguments that represent a pair of conjoined NPs. In Intermediate WML-1 both arguments have the UNITS.LOCATION restriction, and in Intermediate WML-2 only the second argument has that restriction. The NP Conjunction Ambiguity specialist annotates those intermediate WMLs, and the parser proceeds to complete the processing of the input text. In our example, two final WMLs are generated, one for each of two SETOF expressions that originated from the same NP-CONJUNCTION-AMBIGUITY source:

```
WML-1:  (BRING-ABOUT
          ((INTENSION
              (EXISTS ?JX18 LIST
                (OBJECT.OF ?JX18
                   <Interm-WML-1>)))
           TIME WORLD))

WML-2:  (BRING-ABOUT
          ((INTENSION
           (EXISTS ?JX18 LIST
             (OBJECT.OF ?JX18
                <Interm-WML-2>)))
           TIME WORLD))
```

```
ANNOTATION:
(NP-CONJUNCTION-AMBIGUITY
  (Parse-Path-1
    Interps  (WML-1<Interm-WML-1>))
  (Parse-Path-2
    Interps  (WML-2<Interm-WML-2>)))
```

More complex sentences that contain postmodified NP conjunction may have additional interpretations. For instance, the sentence *The carriers were destroyed by frigates and subs in the Indian Ocean* may have a third interpretation in which the PP *in the Indian Ocean* modifies the whole clause. Another more complex example is: *The carriers were destroyed by 3 frigates and subs in the Indian Ocean*, in which ambiguity specialists for NP conjunction, PP clause attachment and quantifier scoping will interact. This kind of interaction among specialists is a topic for our current research on effective paraphrasing.

7.4.2 Phase 2: Translating from WML to Text Structure

Once the Problem Recognizers have annotated the WML, the text planner takes over to translate the intensional logic expression into the hierarchical text structure which organizes the objects and relations specified. In this example, since the input was ambiguous and there are two WMLs, there are two possible strategies for paraphrasing which apply at this step:

(1) Paraphrase of each interpretation separately (as discussed in Section 5.2).

(2) Combine them into a single paraphrase using formatting and highlighting to contrast the differences:

Display **the carriers in the Indian Ocean and the frigates in the Indian Ocean**

or **the carriers in the Indian Ocean and all the frigates.**

We will focus here on the second strategy,

that which combines the interpretations. The text planner will begin by translating one of the WMLS and when it reaches the subexpression that is annotated as being ambiguous, it will build a text structure object representing the disjunction of those subexpressions.

As discussed in Section 5.3.2, the translation to text structure uses both explicit and implicit information from the WML. In this case, the translation of first operator, bring-about builds a complex-event object marked as a command in the present tense and the agent is set to *you*. The domain model concept LIST provides the matrix verb (see text structure in Figure 11).

When the translation reaches the setof expression, a coordinate-relation object is built containing both subexpressions with the relation disjunction. It is also annotated "emphasize-contrast" to guide the later decision making. As this node and its children are expanded, the annotation is passed down. When the translation reaches the individual conjuncts in the expression, it uses the annotation to decide how to expand the text structure for that object. In the case where the modifier distributes, the annotation blocks any optimization and ensures both conjuncts will be modified; in the case where it does not distribute, there are two possible strategies to eliminate the ambiguity:[14]

1) Manipulating the order of the conjuncts in the text structure:

- If only one of the conjuncts is modified and the modifier is realizable as a premodifier, then that conjunct should be placed second.
- If only one of the conjuncts is modified and the modifier is realizable as a postmodifier, then that conjunct should be placed first.

In this case, the paraphrase would be: *Display the frigates in the Indian Ocean and carriers.*

2) Adding a quantifer, such as "all", to the conjunct without modification by adding an adjunct DO to the second conjunct, which would result in the paraphrase: Display all the carriers and the frigates in the Indian Ocean.

We use a combination of these strategies. The figure below shows the text stucture built for this expression[15].

Once this level is complete, it is traversed and the linguistic resources, such as the lexical heads and major syntactic categories, are chosen and represented in the input specification to the lingusitic realization component, Mumble-86, which produces the final text.

7.5 Using the Paraphraser in a Cooperative Dialog System

The work presented here has focused on developing strategies for paraphrasing in order to resolve ambiguity. However, in an actual NL dialog system, choosing when and how to use this capability can be based on other considerations. In this section we address some practical issues and some related

[14]Note that in this task of paraphrasing queries, where it is crucial that the paraphrase be unambiguous, these are strategies the generator should apply regardless of whether the original was ambiguous or not, as ambiguity may have been introduced into a conjunction by some other strategy, such as lexical choice.

[15]Objects labeled DO in the diagram indicate discourse objects which have been created for this utterance. Objects labeled DM are objects from the domain model. The creation of discourse objects allows objects to be annotated with their roles and other information not contained in the domain model (tense, number) and introduces objects which can be referred back to anaphorically with pronouns (e.g. "they for the DO dominating the conjuncts).

work we have done in the integration of our paraphraser into a Man-Machine Interface.

The presentation of a paraphrase can be useful even in cases where no ambiguity has been detected, as it allows the user to verify that the system's interpretation does not differ from the intended interpretation. This is particularly useful for new users who need to be reassured of the system's performance. This feature should be under the user's control, though, since frequent users of the system may only want to see paraphrases when the system finds multiple interpretations.

Paraphrasing can also be incorporated in cooperative responses in order to make any presuppositions explicit. Consider the following exchange:

U: Display all the carriers.
S: <icons displayed on map>
U: Which are within 500 miles of Hawaii?
S: Carriers Midway, Coral Sea, and Saratoga.
U: Which have the highest readiness ratings?
S: Of the carriers within 500 miles of Hawaii, Midway and Saratoga are C1.

Incorporating elided elements from previous queries in the response makes clear which set is being considered for the current answer.

Another sort of paraphrase, which we term "diagnostic responses", can be used when the system is unable to find any interpretation of the user's query, whether due to ill-formedness, novel use of language, or simply inadequate information in the underlying program. As in paraphrasing, the generator uses structures built by the understanding component to generate a focused response. For example, a metaphorical use of "commander" to refer to ships, as in the following query will violate the semantic restrictions on the arguments to the verb "assign". When IRUS-II fails to find a semantic interpretation, it saves its state, which can then be used by the generator to produce an appropriate response:
Q: *Which commanders are assigned to SPA 2?*
S: *I don't understand how commanders can be assigned.*

7.6 Comparison with Other Work

A similar approach to ours is McKeown's Co-op system (McKeown, 1983). It too functions in an interactive environment. However, it is limited in several ways:

1. Since the system it worked with was limited to data base queries, it could only paraphrase questions. This is not only a limitation in functionality, but affects the linguistic competence as well: the input had to be simple WH questions with SVO structure, no complex sentences or complicated adjuncts.

2. It had only one strategy to change the text: given and new[16], which fronted noun phrases with relative clauses or prepositional phrases that appeared in the later parts of the sentence (essentially the verb phrase). For example *Which programmers worked on oceanography projects in 1972?* would be paraphrased: *Assuming that there were oceanography projects in 1972, which programmers worked on those projects?*

3. Since its only strategy involved complex noun phrases, if there were no complex noun phrases in the query, it would be "paraphrased" exactly as the original.

[16] A related problem is that its notion of given and new was very simplistic: it is purely based on syntactic criteria of the incoming sentence and does not consider other criteria such as definiteness of context.

Lowden and de Roeck (1985) also address the problem of paraphrasing in the context of data base query. However, while they assume some parse of a query has taken place, the work focuses entirely on the generation portion of the problem. In fact, they define paraphrasing as providing a "mapping between an underlying formal representation and an NL text." They discuss in detail how text formatting can improve clarity and a solid underlying linguistic framework (in their case lexical functional grammar) can insure grammaticality. However, while they state that a paraphrase should be unambiguous, they do not address how to recognize when a query is ambiguous or how to generate an unambiguous query.

The BBN Parlance™ NL Interface is one of the most robust NL interfaces in existance. Its paraphraser integrates both the system's conceptual and procedural understanding of NL queries. This approach is based on the observation that users need to be shown the conceptual denotation of a word or phrase (e.g., "clerical employee") with its denotation in the underlying database system (e.g., an employee whose EEO category is 3 or an EE whose job title is "secretary"). Thus, the Parlance paraphrases incorporate references to specific fields and values in the underlying data base system. The structure of the paraphrased text closely resembles the strucuture of the interpretation of the query. So, while the text can be cumbersome, it has the advantage of more directly capturing what the system understood. Due to efficiency considerations and limitations on the space for output, the Parlance paraphraser presents the paraphases one at a time, allowing the user to confirm or reject the current interpretation, rather than presenting all paraphrases at the same time. The system allows the user to refer back to previously presented interpretations, but as is the case with the other paraphrasers, related interpretations are not contrasted.

7.7 Conclusion

In addition to being useful in current interactive natural language interfaces, the paraphrase task provides an excellent context to explore interesting issues in both natural language understanding and generation as well as paraphrasing itself. In the next phase of our research we plan to look at quantifier scope ambiguities, lexical choice, and the interaction between multiple problems and strategies for improvement.

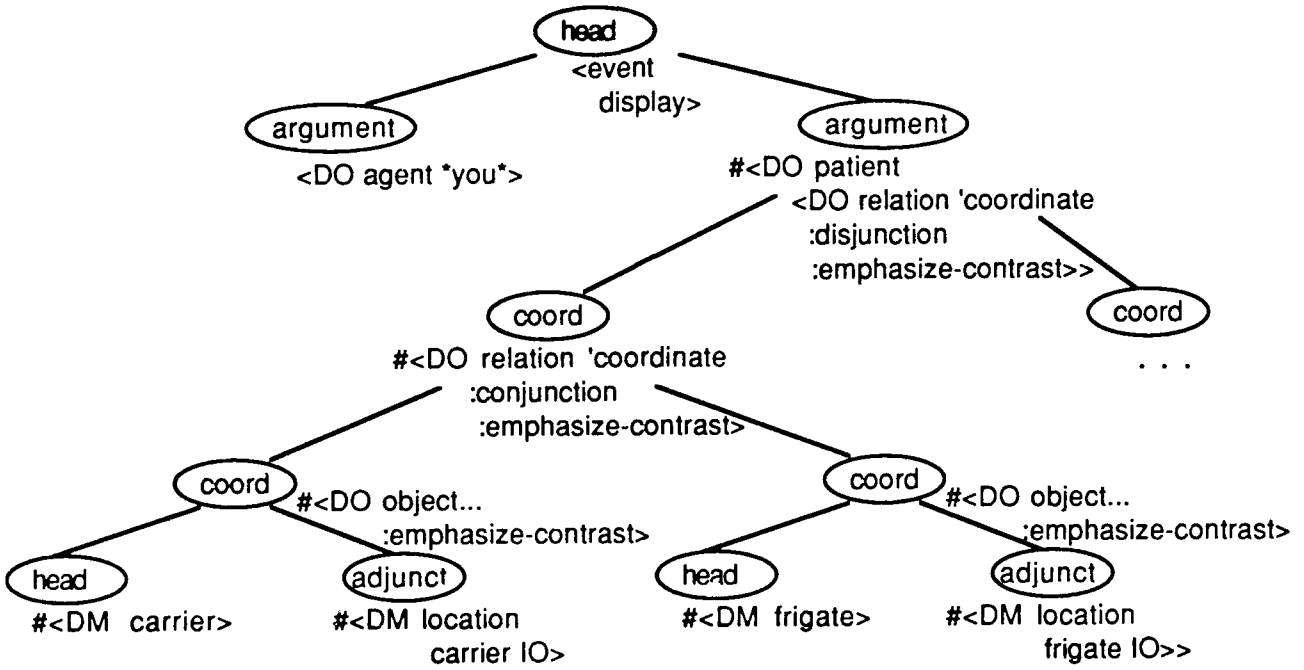

Figure 17: Text Structure for Generation

References

Hinrichs, Erhard, Damaris Ayuso, Remko Scha (1987) "The Syntax and Semantic of the JANUS Semantic Interpretation Language", Technical Report Section of BBN Report No. 6522, BBN Laboratories, pgs. 27-33.

Lowden, Barry G. T., and Anne De Roeck (1985) "Generating English Paraphrases from Relational Query Expressions", vol. 4, no. 4, p.337-348.

McKeown, Kathleen R. (1983) "Paraphrasing Questions Using Given and New Information", American Journal of Computational Linguistics, vol. 9. no. 1, Jan-Mar 1983, p.1-10.

McDonald, David D. (1983) "Description Directed Control", Computers and Mathematics 9(1), Reprinted in Grosz, et al. (eds.), Readings in Natural Language Processing, Morgan Kaufmann Publishers, California, 1986, p. 519-538.

Meteer, Marie M. David D. McDonald, Scott Anderson, David Forster, Linda Gay, Alison Heuttner, Penelope Sibun (1987) Mumble-86: Design and Implementation University of Massachusetts Technical Report 87-87, 173 pages.

Mosher, Margaret (1983) "An Overview of NIKL", Technical Report Section of BBN Report No. 5421, BBN Laboratories.

Weischedel, Ralph, Edward Walker, Damaris Ayuso, Jos de Bruin, Kimberle Koile, Lance Ramshaw, Varda Shaked (1986) "Out of the Laboratory: A case study with the irus natural language interface", in Research and Development in Natural Language Understanding as part of the Strategic Computing Program, BBN Labs Technical Report number 6463, pgs. 13-26.

Weischedel, Ralph, D. Ayuso, A. Haas, E. Hinrichs, R. Scha, V. Shaked (1987) Research and Development in Natural Language Understanding as part of the Strategic Computing Program, BBN Labs Technical Report number 6522.